```
                        PAGE    118,121
                        TITLE   Victor V86P Hard Disk Controller BIOS (2793VG.10010688)

                        ;--------------------------------------------------------
                        ;       INT 13H Commands                                :
                        ;--------------------------------------------------------

= 0000                  INT13H_OP_00_RESET_DISK_SYSTEM          EQU     00h
= 0001                  INT13H_OP_01_GET_DISK_STATUS            EQU     01h
= 0008                  INT13H_OP_08_GET_DRIVE_PARAMS           EQU     08h
= 0009                  INT13H_OP_09_INITIALIZE_DISK_TABLE      EQU     09h
= 000D                  INT13H_OP_0D_ALTERNATE_DISK_RESET       EQU     0Dh
= 0010                  INT13H_OP_10_TEST_DRIVE_READY           EQU     10h
= 0011                  INT13H_OP_11_RECALIBRATE                EQU     11h
= 0012                  INT13H_OP_12_SRAM_DIAGS                 EQU     12h
= 0014                  INT13H_OP_14_CONTROLLER_DIAGS           EQU     14h
=                       LAST_INT13H_OP  EQU     INT13H_OP_14_CONTROLLER_DIAGS


                        ;--------------------------------------------------------
                        ;       INT 13H Errors (returned in AH)                 :
                        ;--------------------------------------------------------

= 0000                  INT13H_STATUS_00_NO_ERROR               EQU     000h
= 0001                  INT13H_STATUS_01_BAD_COMMAND            EQU     001h
= 0002                  INT13H_STATUS_02_ADDR_MARK_NOT_FOUND    EQU     002h
= 0004                  INT13H_STATUS_04_SECTOR_NOT_FOUND       EQU     004h
= 0007                  INT13H_STATUS_07_BAD_DISK_PARAM_TABLE   EQU     007h
= 0009                  INT13H_STATUS_09_DMA_ACROSS_64K         EQU     009h
= 000A                  INT13H_STATUS_0A_BAD_SECTOR_FLAG        EQU     00Ah
= 000B                  INT13H_STATUS_0B_BAD_CYLINDER           EQU     00Bh
= 0010                  INT13H_STATUS_10_ECC_ERROR              EQU     010h
= 0011                  INT13H_STATUS_11_ECC_FIXED              EQU     011h
= 0020                  INT13H_STATUS_20_CTRLR_ERROR            EQU     020h
= 0027                  INT13H_STATUS_27_NEED_RECALIBRATE       EQU     027h    ; Guess. Undocumented.
= 0028                  INT13H_STATUS_28_UNKNOWN_ERROR          EQU     028h    ; Unknown bad sector error.
= 0040                  INT13H_STATUS_40_SEEK_FAILURE           EQU     040h
= 0080                  INT13H_STATUS_80_TIMEOUT                EQU     080h
= 00BB                  INT13H_STATUS_BB_UNDEFINED_ERROR        EQU     0BBh
= 00FF                  INT13H_STATUS_FF_SENSE_OP_FAILED        EQU     0FFh


                        ;--------------------------------------------------------
                        ;       Interrupt vector table (at 0000:0000)           :
                        ;       and BIOS Data Area (from 0000:0400)             :
                        ;--------------------------------------------------------

0000                    ZEROSEG                         SEGMENT AT 0000H

004C                                                    ORG     013H*4
004C ????               INT13H_OFFSET                   DW      ?       ; The disk services. We hook this
004E ????               INT13H_SEGMENT                  DW      ?       ;  and move the original handler over
                                                                        ;  to INT40H.

0064                                                    ORG     019H*4
0064 ????               INT19H_OFFSET                   DW      ?       ; Boot service. We hook this in order
0066 ????               INT19H_SEGMENT                  DW      ?       ;  to try reading and running the
                                                                        ;  hard disk boot sector first, then
                                                                        ;  falling back to the floppy.

0100                                                    ORG     040H*4
0100 ????               INT40H_OFFSET                   DW      ?       ; The original (floppy) disk service
0102 ????               INT40H_SEGMENT                  DW      ?       ;  is relocated here.

0104                                                    ORG     041H*4
0104 ????               INT41H_OFFSET                   DW      ?       ; Data pointer, not code. Points to
0106 ????               INT41H_SEGMENT                  DW      ?       ;  disk geometry and parameters.

0442                                                    ORG     0442h
0442 ??                 BDA_CONTROLLER_DATA_BUFFER_00   DB      ?       ; This is the buffer used for
0443 ??                 BDA_CONTROLLER_DATA_BUFFER_01   DB      ?       ;  constructing commands for submission
0444 ??                 BDA_CONTROLLER_DATA_BUFFER_02   DB      ?       ;  into the disk controller, but also
0445 ??                 BDA_CONTROLLER_DATA_BUFFER_03   DB      ?       ;  for reading the sense data on error.
0446 ??                 BDA_CONTROLLER_DATA_BUFFER_04   DB      ?       ; When a INT13H function returns,
0447 ??                 BDA_CONTROLLER_DATA_BUFFER_05   DB      ?       ;  the first byte here allways holds
0448 ??                 BDA_CONTROLLER_DATA_BUFFER_06   DB      ?       ;  the success/errror status.

046C                                                    ORG     046Ch
046C ????               BDA_TIMER_COUNTER_LO            DW      ?
046E ????               BDA_TIMER_COUNTER_HI            DW      ?

0472                                                    ORG     0472h
0472 ????               BDA_SOFT_RESET_FLAG             DW      ?
0474 ??                 BDA_LAST_OP_STATUS              DB      ?
0475 ??                 BDA_NUMBER_OF_HARD_DISKS        DB      ?

7C00                                                    ORG     7C00H
7C00                    BOOT_SEC                        LABEL   FAR
7DFE                                                    ORG     7DFEh
7DFE                    BOOT_SIG                        LABEL   FAR

7DFE                    ZEROSEG                         ENDS

0000                    BIOSSEG                         SEGMENT AT 0F000H
FFF0                                                    ORG     0FFF0H
FFF0                    RESET_VEC                       LABEL   FAR
FFF0                    BIOSSEG                         ENDS

```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Variables and buffers used by the formatter     :
                        ;       utility. The segment is allocated by DOS        :
                        ;--------------------------------------------------------

0000                    FORMATTER_HEAP              SEGMENT AT 1000H

00B0                                                ORG     00B0h   ; Some terminal I/O
00B0 ??                 SAVED_COLUMN                DB      ?       ;  bookkeeping routines
00B1 ??                 SAVED_ROW                   DB      ?
00B2 ??                 READ_NUMBER_SAVED_AL        DB      ?
00B3 ??                 READ_NUMBER_SAVED_AH        DB      ?
00B4 ????????           PARSE_NUMBER_BUFFER         DD      ?
00B8 ????               READ_NUMBER_RESULT          DW      ?
00BA ??                 READ_NUMBER_RESULT_LEN      DB      ?

01B9                                                ORG     01B9h   ; Formatter variables
01B9 ??                 HARD_DISK_NUMBER            DB      ?
01BA ??                 READ_NUMBER_MIN_DIGITS      DB      ?
01BB ??                 READ_NUMBER_MAX_DIGITS      DB      ?
01BC ??                                             DB      ?
01BD ??                                             DB      ?
01BE ??                 BAD_POINTER                 DB      ?
01BF ??                                             DB      ?
01C0 ??                 CURSOR_LOCATION_OR_SOMETHING DB     ?
01C1 ??                                             DB      ?
01C2 ??                                             DB      ?
01C3 ??                                             DB      ?
01C4 ??                                             DB      ?
01C5 ??                                             DB      ?
01C6 ??                                             DB      ?
01C7 ??                                             DB      ?
01C8 ????               LAST_CYLINDER_NUMBER        DW      ?
01CA ??                                             DB      ?
01CB ??                                             DB      ?
01CC ??                 LAST_HEAD_NUMBER            DB      ?
01CD ??                                             DB      ?
01CE ????               VERIFY_PROGRESS_BUFFER      DW      ?
01D0 ????               VERIFY_PROGRESS_BUFFER_0    DW      ?
01D2 ????               VERIFY_PROGRESS_BUFFER_1    DW      ?
01D4 ??                 UNK_UNUSED_STH              DB      ?
01D5 ??                 CONTROL_BYTE                DB      ?
01D6 ??                 VERIFY_STH_2                DB      ?
01D7 ??                 INTERLEAVE                  DB      ?
01D8 ??                                             DB      ?
01D9 ??                                             DB      ?
01DA ????               CYLINDER_NUMBER             DW      ?
01DC ??                 HEAD_NUMBER                 DB      ?
01DD ??                                             DB      ?
01DE ??                                             DB      ?
01DF ??                                             DB      ?
01E0 ??                                             DB      ?
01E1 ??                                             DB      ?
01E2 ??                                             DB      ?
01E3 ????               CURRENT_BAD_ENTRY           DW      ?
01E5 ??                 CURRENT_BAD_ENTRY_0         DB      ?
01E6 ??                 CYLINDER_TABLE              DB      ?
01E7 ??                                             DB      ?
01E8 ??                                             DB      ?
01E9 ??                                             DB      ?
01EA ??                 BAD_BLOCKS_TABLE_USED_SPACE DB      ?
01EB ??                 NUM_BAD_BLOCK_ENTRIES       DB      ?
01EC ????               UNK_VERIFY_STH              DW      ?
01EE ??                 UNK_VERIFY_STH_0            DB      ?
01EF ????               UNK_VERIFY_UNUSED_1         DW      ?
01F1 ??                                             DB      ?
01F2 ??                                             DB      ?
01F3 ??                                             DB      ?
01F4 ??                                             DB      ?
01F5 ??                                             DB      ?
01F6 ??                                             DB      ?
01F7 ??                                             DB      ?
01F8 ????               UNK_VERIFY_UNUSED_0         DW      ?
01FA ????               UNK_VERIFY_UNUSED           DW      ?


0400                                                ORG     0400h
0400 ??                 BAD_BLOCKS_TABLE            DB      ?

0700                                                ORG     0700h
0700 ??                 SECTOR_BUFFER               DB      ?

0701                    FORMATTER_HEAP              ENDS


```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Controller opcodes                               :
                        ;--------------------------------------------------------

= 0000                  OP_00_TEST_DRIVE_READY                  EQU     000h
= 0001                  OP_01_RECALIBRATE                       EQU     001h
= 0003                  OP_03_READ_SENSE                        EQU     003h
= 0004                  OP_04_FORMAT_DRIVE                      EQU     004h
= 0005                  OP_05_VERIFY_SECTORS                    EQU     005h
= 0006                  OP_06_FORMAT_TRACK                      EQU     006h
= 0007                  OP_07_FORMAT_BAD_TRACK                  EQU     007h
= 0008                  OP_08_READ_SECTORS                      EQU     008h
= 000A                  OP_0A_WRITE_SECTORS                     EQU     00Ah
= 000B                  OP_0B_SEEK                              EQU     00Bh
= 000C                  OP_0C_INIT_DRV_PARM                     EQU     00Ch
= 000D                  OP_0D_READ_ECC_BURST_ERROR_LEN          EQU     00Dh
= 000E                  OP_0E_READ_SECTOR_BUFFER                EQU     00Eh
= 000F                  OP_0F_WRITE_SECTOR_BUFFER               EQU     00Fh
= 00E0                  OP_E0_SECTOR_BUFFER_DIAG                EQU     0E0h
= 00E3                  OP_E3_DRIVE_DIAG                        EQU     0E3h
= 00E4                  OP_E4_CTRL_DIAG                         EQU     0E4h
= 00E5                  OP_E5_READ_LONG                         EQU     0E5h
= 00E6                  OP_E6_WRITE_LONG                        EQU     0E6h


                        ;--------------------------------------------------------
                        ;       Controller I/O ports                             :
                        ;--------------------------------------------------------

= 0320                  IO_PORT_320_DATA                        EQU     320h
= 0321                  IO_PORT_321_READ_STATUS_WRITE_RESET     EQU     321h
= 0322                  IO_PORT_322_READ_CONFIG_WRITE_SELECT    EQU     322h
= 0323                  IO_PORT_323_DMA_IRQ                     EQU     323h


                        ;--------------------------------------------------------
                        ;       DMA mode register                                :
                        ;                                                        :
                        ;       channel=3 | address increment                    :
                        ;       | demand mode | read/write                       :
                        ;--------------------------------------------------------

= 0047                  DRQ3_READ                               EQU     47H
= 004B                  DRQ3_WRITE                              EQU     4BH


                        ;--------------------------------------------------------
                        ;       Main ROM image                                   :
                        ;--------------------------------------------------------

0000                    ROM             SEGMENT
                                        ASSUME  cs:ROM

0000 AA55               MAGIC           DW      0AA55h
0002 10                 LENGTH          DB      16                      ; ROM length in 16-byte paragraphs

0003 E9 0A17 R                          jmp     near ptr ROM_INIT

0015                                    ORG     015h
0015 EB 19                              jmp     short FORMATTER
0017 90                                 nop

0020                                    ORG     020h
0020                    INT41H_DATA:
0020 0267                               MAX_CYLS        DW      615
0022 04                                 MAX_HEADS       DB      4
0023 0064                               RWC             DW      100     ; Cylinder up to which reduced
                                                                        ;  write current is used
0025 015E                               WPC             DW      350     ; Cylinder from which write
                                                                        ;  precompensation is used
0027 07                                 ECC_LEN         DB      7
0028 01                                 OPT_FLAGS       DB      1
0029 04                                 TIMEOUT_STD     DB      4
002A 09                                 TIMEOUT_FMT     DB      9
002B 09                                 TIMEOUT_CHK     DB      9
002C 00 00 00 00                        _RESERVED       DD      0
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Formatter utility                               :
                        ;                                                       :
                        ;       Run it from PC DOS:                             :
                        ;       >DEBUG                                          :
                        ;       -G=C800:15                                      :
                        ;                                                       :
                        ;       Note the disasembly of the formatter has        :
                        ;       not been paid much attention to and the         :
                        ;       quality probably reflects that.                 :
                        ;--------------------------------------------------------

0030                    FORMATTER               PROC NEAR
0030 1E                                 push    ds
0031 07                                 pop     es
0032 0E                                 push    cs
0033 1F                                 pop     ds
                                        ASSUME  es:FORMATTER_HEAP, ds:ROM
0034 BA 07BC R                          mov     dx, offset STRING_FORMAT_WILL_DESTROY
0037 E8 0755 R                          call    PRINT_CRLF_STRING
003A E8 04D5 R                          call    ASK_CONFIRMATION
003D 73 03                              jnc     short READ_DRIVE_NUMBER
003F E9 00F3 R                          jmp     ERROR_EXIT

0042                    READ_DRIVE_NUMBER:
0042 BA 07F4 R                          mov     dx, offset STRING_ENTER_DRIVE
0045 E8 0755 R                          call    PRINT_CRLF_STRING
0048 26: C6 06 01BB R 01                mov     es:READ_NUMBER_MAX_DIGITS, 1
004E 26: C6 06 01BA R 01                mov     es:READ_NUMBER_MIN_DIGITS, 1
0054 E8 0527 R                          call    READ_NUMBER
0057 26: A1 00B8 R                      mov     ax, es:READ_NUMBER_RESULT
005B 3C 00                              cmp     al, 0
005D 74 09                              je      short GOT_VALID_DRIVE_NUMBER
005F 3C 01                              cmp     al, 1
0061 74 05                              je      short GOT_VALID_DRIVE_NUMBER
0063 E8 0765 R                          call    DO_BEEP
0066 EB DA                              jmp     short READ_DRIVE_NUMBER

0068                    GOT_VALID_DRIVE_NUMBER:
0068 26: A2 01B9 R                      mov     es:HARD_DISK_NUMBER, al
006C 3C 00                              cmp     al, 0
006E 75 00                              jne     short $+2
0070 EB 01                              jmp     short A_BIT_FORWARD             ; Why...
0072 90                                 nop
0073 E8 03DA R          A_BIT_FORWARD:  call    READ_DRIVE_PARAMS_AND_MORE
0076 EB 01                              jmp     short YET_FURTHER
0078 90                                 nop
0079 E8 00F8 R          YET_FURTHER:    call    READ_BAD_BLOCKS_TABLE
007C 73 03                              jnc     short GOOD_BAD_BLOCKS_TABLE
007E EB 73                              jmp     short ERROR_EXIT
0080 90                                 nop

0081                    GOOD_BAD_BLOCKS_TABLE:
0081 BF 0700 R                          mov     di, offset SECTOR_BUFFER
0084 8B DF                              mov     bx, di
0086 BA 0080                            mov     dx, 80h
0089 26: A0 01B9 R                      mov     al, es:HARD_DISK_NUMBER
008D 0A D0                              or      dl, al
008F B9 0066                            mov     cx, 102                 ; 510 / 5
0092 FC                                 cld

0093                    FILL_SECTOR_BUFFER:                             ; First 510 bytes
0093 B8 31C6                            mov     ax, 31C6h
0096 26: 89 05                          mov     es:[di], ax
0099 83 C7 02                           add     di, 2
009C B8 638C                            mov     ax, 638Ch
009F 26: 89 05                          mov     es:[di], ax
00A2 83 C7 02                           add     di, 2
00A5 B0 18                              mov     al, 18h
00A7 26: 88 05                          mov     es:[di], al
00AA 47                                 inc     di
00AB E2 E6                              loop    FILL_SECTOR_BUFFER
00AD B8 31C6                            mov     ax, 31C6h               ; Last 2 bytes to 512
00B0 26: 89 05                          mov     es:[di], ax

00B3 B8 0F01                            mov     ax, 0F01h               ; AH=0F Write sector buffer, AL=01 One sector
00B6 B9 0001                            mov     cx, 1
00B9 CD 13                              int     13h
00BB 80 FC 00                           cmp     ah, 0
00BE 74 09                              je      short WRITE_SUCCEEDED
00C0 BA 09A0 R                          mov     dx, offset STRING_BEEP_SECTOR_WRITE_FAIL
00C3 E8 0707 R                          call    PRINT_MESSAGE_AND_DUMP_CONTROLLER_BUFFER
00C6 EB 1C                              jmp     short PROCEED_REBOOTING
00C8 90                                 nop

00C9                    WRITE_SUCCEEDED:
00C9 E8 04F6 R                          call    READ_INTERLEAVE
00CC BA 091F R                          mov     dx, offset STRING_CONFIRM_FORMAT
00CF E8 0755 R                          call    PRINT_CRLF_STRING
00D2 E8 0210 R                          call    READ_YES_OR_NO
00D5 73 03                              jnc     short START_FORMATTING
00D7 EB 1A                              jmp     short ERROR_EXIT
00D9 90                                 nop

00DA                    START_FORMATTING:
00DA E8 0425 R                          call    DO_FORMAT
00DD 72 05                              jc      short PROCEED_REBOOTING
00DF E8 030F R                          call    DO_FORMAT_BAD
00E2 72 00                              jc      short $+2
00E4                    PROCEED_REBOOTING:
00E4 BA 0981 R                          mov     dx, offset STRING_ANY_KEY_TO_REBOOT
00E7 E8 0755 R                          call    PRINT_CRLF_STRING
00EA B4 08                              mov     ah, 8                   ; Keyboard input
00EC CD 21                              int     21h
00EE EA FFF0 ---- R                     jmp     RESET_VEC
00F3                    ERROR_EXIT:
00F3 B8 4C00                            mov     ax, 4C00h
00F6 CD 21                              int     21h
00F8                    FORMATTER               ENDP
```

```
                        PAGE
00F8                    READ_BAD_BLOCKS_TABLE   PROC NEAR
00F8 26: C6 06 01EA R 00                mov     es:BAD_BLOCKS_TABLE_USED_SPACE, 0
00FE 26: C6 06 01EB R 00                mov     es:NUM_BAD_BLOCK_ENTRIES, 0
0104 BA 08B5 R                          mov     dx, offset STRING_ANY_DEFECTS
0107 E8 0755 R                          call    PRINT_CRLF_STRING
010A E8 0210 R                          call    READ_YES_OR_NO
010D 73 03                              jnc     short DO_READ_DEFECTS
010F E9 01BB R                          jmp     DEFECTS_READ_DONE

0112                    DO_READ_DEFECTS:
0112 BA 08CA R                          mov     dx, offset STRING_PRESS_RET_TO_END
0115 E8 0755 R                          call    PRINT_CRLF_STRING

0118                    READ_CYLINDER:

0118 BA 08EC R                          mov     dx, offset STRING_CYLINDER
011B E8 0755 R                          call    PRINT_CRLF_STRING
011E E8 0795 R                          call    MOVE_CURSOR
0121 26: C6 06 01BB R 04                mov     es:READ_NUMBER_MAX_DIGITS, 4
0127 26: C6 06 01BA R 02                mov     es:READ_NUMBER_MIN_DIGITS, 2
012D E8 0527 R                          call    READ_NUMBER
0130 26: 80 3E 00BA R 00                cmp     es:READ_NUMBER_RESULT_LEN, 0    ; Empty input?
0136 75 03                              jne     short CYLINDER_WAS_READ
0138 EB 6D                              jmp     short ENTRY_READ_DONE
013A 90                                 nop

013B                    CYLINDER_WAS_READ:
013B 26: A1 00B8 R                      mov     ax, es:READ_NUMBER_RESULT
013F 26: 39 06 01C8 R                   cmp     es:LAST_CYLINDER_NUMBER, ax
0144 77 05                              ja      short CYLINDER_CHECK_SUCCESSFUL
0146 E8 0780 R                          call    BAD_ENTRY_BEEP
0149 EB CD                              jmp     short READ_CYLINDER


014B                    CYLINDER_CHECK_SUCCESSFUL:
014B 26: A3 01DA R                      mov     es:CYLINDER_NUMBER, ax

014F                    READ_HEAD:
014F BA 08FB R                          mov     dx, offset STRING_HEAD
0152 E8 0755 R                          call    PRINT_CRLF_STRING
0155 E8 0795 R                          call    MOVE_CURSOR
0158 26: C6 06 01BB R 02                mov     es:READ_NUMBER_MAX_DIGITS, 2
015E 26: C6 06 01BA R 01                mov     es:READ_NUMBER_MIN_DIGITS, 1
0164 E8 0527 R                          call    READ_NUMBER
0167 26: A1 00B8 R                      mov     ax, es:READ_NUMBER_RESULT
016B 26: 38 06 01CC R                   cmp     es:LAST_HEAD_NUMBER, al
0170 76 19                              jbe     short BAD_HEAD_READ_AGAIN
0172 26: 8B 0E 01DA R                   mov     cx, es:CYLINDER_NUMBER
0177 41                                 inc     cx
0178 26: 3B 0E 01C8 R                   cmp     cx, es:LAST_CYLINDER_NUMBER
017D 72 11                              jb      short HEAD_READ_SUCCESSFUL
017F 26: 8A 26 01CC R                   mov     ah, es:LAST_HEAD_NUMBER
0184 2A E0                              sub     ah, al
0186 80 FC 01                           cmp     ah, 1
0189 77 05                              ja      short HEAD_READ_SUCCESSFUL

018B                    BAD_HEAD_READ_AGAIN:
018B E8 0780 R                          call    BAD_ENTRY_BEEP
018E EB BF                              jmp     short READ_HEAD

0190                    HEAD_READ_SUCCESSFUL:
0190 26: A2 01DC R                      mov     es:HEAD_NUMBER, al
0194 E8 01CC R                          call    PROCESS_BAD_BLOCKS_ENTRY
0197 26: FE 06 01EB R                   inc     es:NUM_BAD_BLOCK_ENTRIES
019C 26: 80 3E 01EB R C0                cmp     es:NUM_BAD_BLOCK_ENTRIES, 0C0h
01A2 73 11                              jnb     short DEFECTS_TABLE_FULL
01A4 E9 0118 R                          jmp     READ_CYLINDER

01A7                    ENTRY_READ_DONE:
01A7 BA 090A R                          mov     dx, offset STRING_ASK_MORE_DEFECT_ENTRIES
01AA E8 0755 R                          call    PRINT_CRLF_STRING
01AD E8 0210 R                          call    READ_YES_OR_NO
01B0 72 09                              jc      short DEFECTS_READ_DONE
01B2 E9 0118 R                          jmp     READ_CYLINDER

01B5                    DEFECTS_TABLE_FULL:
01B5 BA 089B R                          mov     dx, offset STRING_NO_MORE_DEFECTS
01B8 E8 0755 R                          call    PRINT_CRLF_STRING

01BB                    DEFECTS_READ_DONE:
01BB 26: 80 3E 01EB R 00                cmp     es:NUM_BAD_BLOCK_ENTRIES, 0
01C1 74 05                              je      short SUCCESS
01C3 E8 04D5 R                          call    ASK_CONFIRMATION
01C6 72 02                              jc      short ABORTED

01C8                    SUCCESS:
01C8 F8                                 clc
01C9 C3                                 ret
01CA                    ABORTED:
01CA F9                                 stc
01CB C3                                 ret
01CC                    READ_BAD_BLOCKS_TABLE   ENDP


```

```
                        PAGE
01CC                    PROCESS_BAD_BLOCKS_ENTRY        PROC NEAR
01CC BB 01E6 R                  mov     bx, offset CYLINDER_TABLE
01CF 26: A1 01DA R              mov     ax, es:CYLINDER_NUMBER
01D3 26: 88 47 03               mov     es:[bx+3], al
01D7 80 E4 07                   and     ah, 7
01DA B1 06                      mov     cl, 6
01DC D2 E4                      shl     ah, cl
01DE 73 06                      jnb     short HDD
01E0 26: 80 0E 01DC R 80        or      es:HEAD_NUMBER, 80h
01E6                    HDD:
01E6 26: 88 67 02               mov     es:[bx+2], ah
01EA 26: A0 01DC R              mov     al, es:HEAD_NUMBER
01EE 26: 88 47 01               mov     es:[bx+1], al
01F2 E8 0451 R                  call    FORMAT_BAD_BLOCK_ENTRY
01F5 BB 0400 R                  mov     bx, offset BAD_BLOCKS_TABLE
01F8 33 C9                      xor     cx, cx
01FA 26: 8A 0E 01EA R           mov     cl, es:BAD_BLOCKS_TABLE_USED_SPACE
01FF 03 D9                      add     bx, cx
0201 26: 89 07                  mov     es:[bx], ax
0204 26: 88 57 02               mov     es:[bx+2], dl
0208 26: 80 06 01EA R 04        add     es:BAD_BLOCKS_TABLE_USED_SPACE, 4
020E F8                         clc
020F C3                         ret
0210                    PROCESS_BAD_BLOCKS_ENTRY        ENDP


0210                    READ_YES_OR_NO          PROC NEAR
0210 E8 06D3 R                  call    READ_CURSOR_POS
0213                    READ_ANSWER:
0213 B4 01                      mov     ah, 1                   ; Read keyboard
0215 CD 21                      int     21h
0217 50                         push    ax
0218 B4 08                      mov     ah, 8                   ; Read, no echo
021A CD 21                      int     21h
021C 3C 0D                      cmp     al, 0Dh                 ; <ENTER>
021E 74 09                      je      short CONFIRMED_ENTER
0220 58                         pop     ax
0221                    BAD_ANSWER:
0221 E8 0765 R                  call    DO_BEEP
0224 E8 06EE R                  call    SET_CURSOR_POS
0227 EB EA                      jmp     short READ_ANSWER
0229                    CONFIRMED_ENTER:
0229 58                         pop     ax
022A 24 5F                      and     al, 5Fh
022C 3C 59                      cmp     al, 'Y'
022E 75 02                      jne     short ANSWER_NO
0230 F8                         clc
0231 C3                         ret
0232                    ANSWER_NO:
0232 3C 4E                      cmp     al, 'N'
0234 75 EB                      jne     short BAD_ANSWER
0236 F9                         stc
0237 C3                         ret
0238                    READ_YES_OR_NO          ENDP


0238                    STORE_CURRENT_BAD_ENTRY PROC NEAR
0238 26: A3 01E3 R              mov     es:CURRENT_BAD_ENTRY, ax
023C 26: 88 16 01E5 R           mov     es:CURRENT_BAD_ENTRY_0, dl
0241 C3                         ret
0242                    STORE_CURRENT_BAD_ENTRY ENDP


0242                    LOAD_CURRENT_BAD_ENTRY  PROC NEAR
0242 33 D2                      xor     dx, dx
0244 26: A1 01E3 R              mov     ax, es:CURRENT_BAD_ENTRY
0248 26: 8A 16 01E5 R           mov     dl, es:CURRENT_BAD_ENTRY_0
024D C3                         ret
024E                    LOAD_CURRENT_BAD_ENTRY  ENDP

```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       The Formatter Victor V86P Hard Drive ROM        :
                        ;       has verification neutered. The bad blocks       :
                        ;       had to be entered manually.                     :
                        ;                                                       :
                        ;       The disassembly quality of these unreachable    :
                        ;       routines is thus even lower. Sorry.             :
                        ;--------------------------------------------------------

024E                    UNUSED_VERIFY           PROC NEAR
024E BA 087B R                          mov     dx, offset STRING_VERIFYING
0251 E8 0755 R                          call    PRINT_CRLF_STRING
0254 26: C7 06 01FA R 0000                 mov     es:UNK_VERIFY_UNUSED, 0
025B E8 036D R                          call    UNK_VERIFY_2
025E 33 D2                              xor     dx, dx
0260 33 C0                              xor     ax, ax

0262                    VERIFY_NEXT:
0262 50                                 push    ax
0263 52                                 push    dx

0264 E8 0496 R                          call    CURRENT_BAD_ENTRY_TO_INT13H_PARAMS
0267 B8 0011                            mov     ax, 17          ; Verify 17 sectors
026A B4 04                              mov     ah, 4           ; Verify sectors
026C BB 0200                            mov     bx, 512         ; 512 bytes
026F CD 13                              int     13h

0271 80 FC 00                           cmp     ah, INT13H_STATUS_00_NO_ERROR
0274 74 28                              je      short VERIFY_FOUND_GOOD
0276 80 FC 11                           cmp     ah, INT13H_STATUS_11_ECC_FIXED
0279 74 23                              je      short VERIFY_FOUND_GOOD
027B 80 FC 0B                           cmp     ah, INT13H_STATUS_0B_BAD_CYLINDER
027E 74 1E                              je      short VERIFY_FOUND_GOOD

0280 80 FC 10                           cmp     ah, INT13H_STATUS_10_ECC_ERROR
0283 74 67                              je      short VERIFY_FOUND_BAD
0285 80 FC 02                           cmp     ah, INT13H_STATUS_02_ADDR_MARK_NOT_FOUND
0288 74 62                              je      short VERIFY_FOUND_BAD
028A 80 FC 04                           cmp     ah, INT13H_STATUS_04_SECTOR_NOT_FOUND
028D 74 5D                              je      short VERIFY_FOUND_BAD
028F 80 FC 28                           cmp     ah, INT13H_STATUS_28_UNKNOWN_ERROR
0292 74 58                              je      short VERIFY_FOUND_BAD
0294 5A                                 pop     dx
0295 58                                 pop     ax
0296 BA 0834 R                          mov     dx, offset STRING_VERIFY_FAILED
0299 E8 0707 R                          call    PRINT_MESSAGE_AND_DUMP_CONTROLLER_BUFFER
029C F9                                 stc
029D C3                                 ret

029E                    VERIFY_FOUND_GOOD:
029E 5A                                 pop     dx
029F 58                                 pop     ax
02A0 B9 0011                            mov     cx, 17
02A3 03 C1                              add     ax, cx
02A5 80 D2 00                           adc     dl, 0
02A8 26: 8B 0E 01EC R                   mov     cx, es:UNK_VERIFY_STH
02AD 26: 8A 1E 01EE R                   mov     bl, es:UNK_VERIFY_STH_0
02B2 3A DA                              cmp     bl, dl
02B4 77 06                              ja      short VERIFY_NOT_DONE_YET
02B6 3B C8                              cmp     cx, ax
02B8 77 02                              ja      short VERIFY_NOT_DONE_YET
02BA EB 42                              jmp     short VERIFY_DONE

02BC                    VERIFY_NOT_DONE_YET:
02BC 50                                 push    ax
02BD 52                                 push    dx
02BE 26: A1 01FA R                      mov     ax, es:UNK_VERIFY_UNUSED
02C2 B9 0011                            mov     cx, 11h
02C5 03 C1                              add     ax, cx
02C7 26: A3 01FA R                      mov     es:UNK_VERIFY_UNUSED, ax
02CB 26: 8B 0E 01F8 R                   mov     cx, es:UNK_VERIFY_UNUSED_0
02D0 3B C1                              cmp     ax, cx
02D2 72 13                              jb      short GO_VERIFY_NEXT
02D4 2B C1                              sub     ax, cx
02D6 26: A3 01FA R                      mov     es:UNK_VERIFY_UNUSED, ax
02DA 26: A0 01EF R                      mov     al, byte ptr es:UNK_VERIFY_UNUSED_1
02DE 04 05                              add     al, 5
02E0 26: A2 01EF R                      mov     byte ptr es:UNK_VERIFY_UNUSED_1, al
02E4 E8 039E R                          call    VERIFY_PROGRESS

02E7                    GO_VERIFY_NEXT:
02E7 5A                                 pop     dx
02E8 58                                 pop     ax
02E9 E9 0262 R                          jmp     VERIFY_NEXT

02EC                    VERIFY_FOUND_BAD:
02EC 5A                                 pop     dx
02ED 58                                 pop     ax
02EE E8 0238 R                          call    STORE_CURRENT_BAD_ENTRY
02F1 E8 0350 R                          call    FORMAT_BAD_TRACK
02F4 73 01                              jnc     short BAD_FORMAT_SUCCESSFUL
02F6 C3                                 ret

02F7                    BAD_FORMAT_SUCCESSFUL:
02F7 E8 0242 R                          call    LOAD_CURRENT_BAD_ENTRY
02FA 50                                 push    ax
02FB 52                                 push    dx
02FC EB A0                              jmp     short VERIFY_FOUND_GOOD

02FE                    VERIFY_DONE:
02FE 26: C6 06 01EF R 64                mov     byte ptr es:UNK_VERIFY_UNUSED_1, 64h
0304 E8 039E R                          call    VERIFY_PROGRESS
0307 BA 088A R                          mov     dx, offset STRING_VERIFY_COMPLETE
030A E8 0755 R                          call    PRINT_CRLF_STRING
030D F8                                 clc
030E C3                                 ret
030F                    UNUSED_VERIFY           ENDP
```

```
                        PAGE
030F                    DO_FORMAT_BAD           PROC NEAR
030F 26: 80 3E 01EB R 00                cmp     es:NUM_BAD_BLOCK_ENTRIES, 0
0315 74 37                              je      short FORMAT_BAD_DONE
0317 BA 09EC R                          mov     dx, offset STRING_PROCESSING_DEFECTS
031A E8 0755 R                          call    PRINT_CRLF_STRING
031D BB 0400                            mov     bx, 400h
0320 26: C6 06 01BE R 00                mov     es:BAD_POINTER, 0

0326                    FORMAT_TRACK:
0326 26: A0 01BE R                      mov     al, es:BAD_POINTER
032A 26: 3A 06 01EB R                   cmp     al, es:NUM_BAD_BLOCK_ENTRIES
032F 74 1D                              je      short FORMAT_BAD_DONE
0331 26: 8B 07                          mov     ax, es:[bx]
0334 26: 8A 57 02                       mov     dl, es:[bx+2]
0338 E8 0238 R                          call    STORE_CURRENT_BAD_ENTRY
033B 53                                 push    bx
033C E8 0350 R                          call    FORMAT_BAD_TRACK
033F 73 02                              jnc     short NEXT_TRACK
0341 5B                                 pop     bx
0342 C3                                 ret

0343                    NEXT_TRACK:
0343 5B                                 pop     bx
0344 83 C3 04                           add     bx, 4
0347 26: FE 06 01BE R                   inc     es:BAD_POINTER
034C EB D8                              jmp     short FORMAT_TRACK

034E                    FORMAT_BAD_DONE:
034E F8                                 clc
034F C3                                 ret
0350                    DO_FORMAT_BAD           ENDP


0350                    FORMAT_BAD_TRACK        PROC NEAR
0350 E8 0242 R                          call    LOAD_CURRENT_BAD_ENTRY
0353 E8 0496 R                          call    CURRENT_BAD_ENTRY_TO_INT13H_PARAMS
0356 B4 06                              mov     ah, 6                   ; Format bad track
0358 26: A0 01D7 R                      mov     al, es:INTERLEAVE
035C CD 13                              int     13h
035E 80 FC 00                           cmp     ah, 0
0361 74 08                              je      short BAD_TRACK_FORMAT_SUCCESSFUL
0363 BA 081C R                          mov     dx, offset STRING_BAD_SECTOR_FAILED
0366 E8 0707 R                          call    PRINT_MESSAGE_AND_DUMP_CONTROLLER_BUFFER
0369 F9                                 stc
036A C3                                 ret
036B                    BAD_TRACK_FORMAT_SUCCESSFUL:
036B F8                                 clc
036C C3                                 ret
036D                    FORMAT_BAD_TRACK        ENDP


036D                    UNK_VERIFY_2            PROC NEAR
036D 26: A1 01C8 R                      mov     ax, es:LAST_CYLINDER_NUMBER
0371 33 C9                              xor     cx, cx
0373 26: 8A 0E 01CC R                   mov     cl, es:LAST_HEAD_NUMBER
0378 F7 E1                              mul     cx
037A B9 0011                            mov     cx, 17
037D F7 E1                              mul     cx
037F B9 0014                            mov     cx, 14h
0382 F7 F1                              div     cx
0384 26: A3 01F8 R                      mov     es:UNK_VERIFY_UNUSED_0, ax
0388 26: C6 06 01EF R 00                mov     byte ptr es:UNK_VERIFY_UNUSED_1, 0
038E E8 06D3 R                          call    READ_CURSOR_POS
0391 E8 076C R                          call    LOAD_CURSOR_POS
0394 26: A2 00B2 R                      mov     es:READ_NUMBER_SAVED_AL, al
0398 26: 88 26 00B3 R                   mov     es:READ_NUMBER_SAVED_AH, ah
039D C3                                 ret
039E                    UNK_VERIFY_2            ENDP


039E                    VERIFY_PROGRESS         PROC NEAR
039E 26: A0 00B2 R                      mov     al, es:READ_NUMBER_SAVED_AL
03A2 26: 8A 26 00B3 R                   mov     ah, es:READ_NUMBER_SAVED_AH
03A7 E8 0776 R                          call    STORE_CURSOR_POS
03AA E8 06EE R                          call    SET_CURSOR_POS
03AD 26: A0 01EF R                      mov     al, byte ptr es:UNK_VERIFY_UNUSED_1
03B1 32 E4                              xor     ah, ah
03B3 26: A3 01D6 R                      mov     word ptr es:VERIFY_STH_2, ax
03B7 33 D2                              xor     dx, dx
03B9 E8 0626 R                          call    VERIFY_PROGRESS_STH
03BC BE 01CF R                          mov     si, (offset VERIFY_PROGRESS_BUFFER+1)
03BF B9 0004                            mov     cx, 4
03C2                    PROGRESS_ANOTHER_CHAR:
03C2 26: 80 3C 20                       cmp     byte ptr es:[si], ' '
03C6 75 04                              jne     short PROGRESS_DONE
03C8 46                                 inc     si
03C9 49                                 dec     cx
03CA EB F6                              jmp     short PROGRESS_ANOTHER_CHAR
03CC                    PROGRESS_DONE:
03CC 1E                                 push    ds
03CD 06                                 push    es
03CE 1F                                 pop     ds
                                        ASSUME  ds:FORMATTER_HEAP
03CF E8 0B3C R                          call    PRINT_CX_CHARS_FROM_DS_SI
03D2 1F                                 pop     ds
                                        ASSUME  ds:ZEROSEG
03D3 BA 099C R                          mov     dx, offset STRING_PERCENT
03D6 E8 074C R                          call    PRINT_STRING_NO_CRLF
03D9 C3                                 ret
03DA                    VERIFY_PROGRESS         ENDP


```

```
                        PAGE
03DA                    READ_DRIVE_PARAMS_AND_MORE      PROC NEAR
03DA B4 08                      mov     ah, 8                           ; Get drive parameters
03DC E8 06C5 R                  call    INT13H_FOR_CHOSEN_DRIVE
03DF FE C6                      inc     dh
03E1 26: 88 36 01CC R           mov     es:LAST_HEAD_NUMBER, dh
03E6 8A D1                      mov     dl, cl
03E8 80 E2 C0                   and     dl, 0C0h
03EB B1 06                      mov     cl, 6
03ED D2 EA                      shr     dl, cl
03EF 8A CD                      mov     cl, ch
03F1 8A EA                      mov     ch, dl
03F3 83 C1 02                   add     cx, 2

03F6 51                         push    cx
03F7 B4 FE                      mov     ah, 0FEh                        ; Undocumented disk function
03F9 E8 06C5 R                  call    INT13H_FOR_CHOSEN_DRIVE         ; Perhaps this is pointless?
03FC 59                         pop     cx
03FD 03 CA                      add     cx, dx

03FF 26: 89 0E 01C8 R           mov     es:LAST_CYLINDER_NUMBER, cx
0404 8B C1                      mov     ax, cx
0406 48                         dec     ax
0407 33 D2                      xor     dx, dx
0409 26: 8A 0E 01CC R           mov     cl, es:LAST_HEAD_NUMBER
040E 32 ED                      xor     ch, ch
0410 F7 E1                      mul     cx
0412 FE C9                      dec     cl
0414 03 C1                      add     ax, cx
0416 B9 0011                    mov     cx, 11h
0419 F7 E1                      mul     cx
041B 26: A3 01EC R              mov     es:UNK_VERIFY_STH, ax
041F 26: 88 16 01EE R           mov     es:UNK_VERIFY_STH_0, dl
0424 C3                         ret
0425                    READ_DRIVE_PARAMS_AND_MORE      ENDP


0425                    DO_FORMAT               PROC NEAR
0425 26: C6 06 01EA R 00        mov     es:BAD_BLOCKS_TABLE_USED_SPACE, 0
042B BA 085C R                  mov     dx, offset STRING_FORMATTING
042E E8 0755 R                  call    PRINT_CRLF_STRING
0431 B6 00                      mov     dh, 0
0433 B4 07                      mov     ah, 7                           ; Format drive
0435 26: A0 01D7 R              mov     al, es:INTERLEAVE               ; Interleave
0439 E8 06C5 R                  call    INT13H_FOR_CHOSEN_DRIVE
043C 80 FC 00                   cmp     ah, 0
043F 74 08                      je      short FORMAT_DONE
0441 BA 080D R                  mov     dx, offset STRING_FORMAT_FAILED
0444 E8 0707 R                  call    PRINT_MESSAGE_AND_DUMP_CONTROLLER_BUFFER
0447 F9                         stc
0448 C3                         ret
0449                    FORMAT_DONE:
0449 BA 086B R                  mov     dx, offset STRING_FORMAT_COMPLETE
044C E8 0755 R                  call    PRINT_CRLF_STRING
044F F8                         clc
0450 C3                         ret
0451                    DO_FORMAT               ENDP


0451                    FORMAT_BAD_BLOCK_ENTRY  PROC NEAR
0451 53                         push    bx
0452 51                         push    cx
0453 33 C0                      xor     ax, ax
0455 26: 8A 47 03               mov     al, es:[bx+3]
0459 26: 8A 67 02               mov     ah, es:[bx+2]
045D B1 01                      mov     cl, 1
045F D2 EC                      shr     ah, cl
0461 26: 8A 4F 01               mov     cl, es:[bx+1]
0465 80 E1 80                   and     cl, 80h
0468 0A E1                      or      ah, cl
046A B1 05                      mov     cl, 5
046C D2 EC                      shr     ah, cl
046E 33 C9                      xor     cx, cx
0470 26: 8A 0E 01CC R           mov     cl, es:LAST_HEAD_NUMBER
0475 F7 E1                      mul     cx
0477 26: 8A 4F 01               mov     cl, es:[bx+1]
047B 81 E1 003F                 and     cx, 3Fh
047F 03 C1                      add     ax, cx
0481 B9 0011                    mov     cx, 11h
0484 F7 E1                      mul     cx
0486 26: 8A 4F 02               mov     cl, es:[bx+2]
048A 81 E1 003F                 and     cx, 3Fh
048E 03 C1                      add     ax, cx
0490 83 D2 00                   adc     dx, 0
0493 59                         pop     cx
0494 5B                         pop     bx
0495 C3                         ret
0496                    FORMAT_BAD_BLOCK_ENTRY  ENDP


```

```
                        PAGE
0496                    CURRENT_BAD_ENTRY_TO_INT13H_PARAMS      PROC NEAR
0496 50                         push    ax
0497 52                         push    dx
0498 33 C0                      xor     ax, ax
049A 26: A0 01CC R              mov     al, es:LAST_HEAD_NUMBER
049E B9 0011                    mov     cx, 11h
04A1 F7 E1                      mul     cx
04A3 8B C8                      mov     cx, ax
04A5 5A                         pop     dx
04A6 58                         pop     ax
04A7 F7 F1                      div     cx
04A9 50                         push    ax
04AA 8B C2                      mov     ax, dx
04AC B1 11                      mov     cl, 11h
04AE 32 ED                      xor     ch, ch
04B0 80 E1 3F                   and     cl, 3Fh
04B3 F6 F1                      div     cl
04B5 FE C4                      inc     ah
04B7 8A CC                      mov     cl, ah
04B9 8A F0                      mov     dh, al
04BB 26: 8A 16 01B9 R           mov     dl, es:HARD_DISK_NUMBER
04C0 80 CA 80                   or      dl, 80h
04C3 58                         pop     ax
04C4 8A E8                      mov     ch, al
04C6 80 E4 07                   and     ah, 7
04C9 51                         push    cx
04CA B9 0006                    mov     cx, 6
04CD D2 E4                      shl     ah, cl
04CF 73 00                      jnb     short $+2
04D1 59                         pop     cx
04D2 0A CC                      or      cl, ah
04D4 C3                         ret
04D5                    CURRENT_BAD_ENTRY_TO_INT13H_PARAMS      ENDP


04D5                    ASK_CONFIRMATION        PROC NEAR
04D5 BA 0953 R                  mov     dx, offset STRING_PRESS_RET_TO_PROCEED
04D8 E8 0755 R                  call    PRINT_CRLF_STRING
04DB E8 06D3 R                  call    READ_CURSOR_POS
04DE                    READ_KEY_RESPONSE:
04DE E8 06EE R                  call    SET_CURSOR_POS
04E1 B4 08                      mov     ah, 8                   ; Read key, no echo
04E3 CD 21                      int     21h
04E5 3C 1B                      cmp     al, 1Bh                 ; <ESC>
04E7 75 02                      jne     short NOT_ESC
04E9 F9                         stc
04EA C3                         ret
04EB                    NOT_ESC:
04EB 3C 0D                      cmp     al, 0Dh                 ; <ENTER>
04ED 74 05                      je      short CONFIRMED_WITH_ENTER
04EF E8 0765 R                  call    DO_BEEP
04F2 EB EA                      jmp     short READ_KEY_RESPONSE
04F4                    CONFIRMED_WITH_ENTER:
04F4 F8                         clc
04F5 C3                         ret
04F6                    ASK_CONFIRMATION        ENDP


04F6                    READ_INTERLEAVE         PROC NEAR
04F6 E8 06D3 R                  call    READ_CURSOR_POS
04F9                    READ_INTERLEAVE_NUMBER:
04F9 E8 06EE R                  call    SET_CURSOR_POS
04FC BA 09B5 R                  mov     dx, offset STRING_INTERLEAVE
04FF E8 0755 R                  call    PRINT_CRLF_STRING
0502 26: C6 06 01BB R 02        mov     es:READ_NUMBER_MAX_DIGITS, 2
0508 26: C6 06 01BA R 01        mov     es:READ_NUMBER_MIN_DIGITS, 1
050E E8 0527 R                  call    READ_NUMBER
0511 26: A1 00B8 R              mov     ax, es:READ_NUMBER_RESULT
0515 3C 01                      cmp     al, 1
0517 72 09                      jb      short WRONG_INTERLEAVE_NUMBER
0519 3C 10                      cmp     al, 10h
051B 77 05                      ja      short WRONG_INTERLEAVE_NUMBER
051D 26: A2 01D7 R              mov     es:INTERLEAVE, al
0521 C3                         ret
0522                    WRONG_INTERLEAVE_NUMBER:
0522 E8 0765 R                  call    DO_BEEP
0525 EB D2                      jmp     short READ_INTERLEAVE_NUMBER
0527                    READ_INTERLEAVE         ENDP


```

```
                        PAGE
0527                    READ_NUMBER             PROC NEAR
0527 E8 06D3 R                          call    READ_CURSOR_POS
052A E8 076C R                          call    LOAD_CURSOR_POS
052D 26: 88 26 00B3 R                   mov     es:READ_NUMBER_SAVED_AH, ah
0532 26: A2 00B2 R                      mov     es:READ_NUMBER_SAVED_AL, al
0536 BB 00B4 R                          mov     bx, offset PARSE_NUMBER_BUFFER
0539 B9 0000                            mov     cx, 0
053C 26: C6 06 01C0 R 00                mov     es:CURSOR_LOCATION_OR_SOMETHING, 0
0542                    READ_NEXT_KEY:

0542 B4 01                              mov     ah, 1                           ; Read key
0544 CD 21                              int     21h
0546 3C 0D                              cmp     al, 0Dh                         ; <ENTER>
0548 74 3F                              je      short ENTER_PRESSED
054A 3C 08                              cmp     al, 8
054C 75 10                              jne     short NOT_BACKSPACE
054E E8 065C R                          call    BACKSPACE_WIPE_WITH_SPACE
0551 26: FE 0E 01C0 R                   dec     es:CURSOR_LOCATION_OR_SOMETHING ; Move cursor back
0556 4B                                 dec     bx
0557 83 E9 01                           sub     cx, 1
055A 72 5F                              jb      short BAD_INPUT                 ; Reached column -1?
055C EB E4                              jmp     short READ_NEXT_KEY
055E                    NOT_BACKSPACE:
055E 26: 3A 0E 01BB R                   cmp     cl, es:READ_NUMBER_MAX_DIGITS
0563 74 56                              je      short BAD_INPUT
0565 26: 88 07                          mov     es:[bx], al
0568 26: A0 01C0 R                      mov     al, es:CURSOR_LOCATION_OR_SOMETHING
056C FE C0                              inc     al
056E 26: A2 01C0 R                      mov     es:CURSOR_LOCATION_OR_SOMETHING, al
0572 26: 80 3E 01BA R 03                cmp     es:READ_NUMBER_MIN_DIGITS, 3
0578 75 0B                              jne     short NEXT_CHARACTER
057A 26: 80 3E 01C0 R 01                cmp     es:CURSOR_LOCATION_OR_SOMETHING, 1
0580 75 03                              jne     short NEXT_CHARACTER
0582 E8 05DC R                          call    UPDATE_CURSOR_AND_WIPE
0585                    NEXT_CHARACTER:
0585 43                                 inc     bx
0586 41                                 inc     cx
0587 EB B9                              jmp     short READ_NEXT_KEY
0589                    ENTER_PRESSED:
0589 26: A0 01C0 R                      mov     al, es:CURSOR_LOCATION_OR_SOMETHING
058D 26: A2 00BA R                      mov     es:READ_NUMBER_RESULT_LEN, al
0591 26: 80 3E 01BA R 01                cmp     es:READ_NUMBER_MIN_DIGITS, 1
0597 74 0B                              je      short ONE_DIGIT
0599 26: 80 3E 01C0 R 00                cmp     es:CURSOR_LOCATION_OR_SOMETHING, 0
059F 74 18                              je      short ALL_GOOD
05A1 EB 0C                              jmp     short MORE_THAN_ONE_DIGIT
05A3 90                                 nop
05A4                    ONE_DIGIT:
05A4 26: 80 3E 01C0 R 00                cmp     es:CURSOR_LOCATION_OR_SOMETHING, 0
05AA 75 03                              jne     short MORE_THAN_ONE_DIGIT
05AC EB 0D                              jmp     short BAD_INPUT
05AE 90                                 nop
05AF                    MORE_THAN_ONE_DIGIT:
05AF E8 05EE R                          call    PARSE_NUMBER
05B2 72 07                              jc      short BAD_INPUT
05B4 26: 89 0E 00B8 R                   mov     es:READ_NUMBER_RESULT, cx
05B9                    ALL_GOOD:
05B9 F8                                 clc
05BA C3                                 ret
05BB                    BAD_INPUT:
05BB E8 0765 R                          call    DO_BEEP
05BE 26: C6 06 01C0 R 00                mov     es:CURSOR_LOCATION_OR_SOMETHING, 0
05C4 26: 8A 26 00B3 R                   mov     ah, es:READ_NUMBER_SAVED_AH
05C9 26: A0 00B2 R                      mov     al, es:READ_NUMBER_SAVED_AL
05CD E8 0776 R                          call    STORE_CURSOR_POS
05D0 E8 05E1 R                          call    SET_CURSOR_POS_AND_WIPE
05D3 B9 0000                            mov     cx, 0
05D6 BB 00B4 R                          mov     bx, offset PARSE_NUMBER_BUFFER
05D9 E9 0542 R                          jmp     READ_NEXT_KEY
05DC                    READ_NUMBER             ENDP


05DC                    UPDATE_CURSOR_AND_WIPE  PROC NEAR
05DC E8 06D3 R                          call    READ_CURSOR_POS
05DF EB 03                              jmp     short WIPE_THINGS_WITH_SPACES
05E1                    UPDATE_CURSOR_AND_WIPE  ENDP


05E1                    SET_CURSOR_POS_AND_WIPE PROC NEAR
05E1 E8 06EE R                          call    SET_CURSOR_POS
05E4                    SET_CURSOR_POS_AND_WIPE ENDP


05E4                    WIPE_THINGS_WITH_SPACES PROC NEAR
05E4 BA 0948 R                          mov     dx, offset STRING_SPACES
05E7 E8 074C R                          call    PRINT_STRING_NO_CRLF
05EA E8 06EE R                          call    SET_CURSOR_POS
05ED C3                                 ret
05EE                    WIPE_THINGS_WITH_SPACES ENDP


```

```
                        PAGE
05EE                    PARSE_NUMBER            PROC NEAR
05EE BB 00B4 R                          mov     bx, offset PARSE_NUMBER_BUFFER
05F1 33 C9                              xor     cx, cx
05F3 26: 80 3E 01C0 R 00                cmp     es:CURSOR_LOCATION_OR_SOMETHING, 0
05F9 74 27                              je      short LAST_DIGIT
05FB                    NEXT_DIGIT:
05FB 26: 8A 07                          mov     al, es:[bx]
05FE 3C 30                              cmp     al, '0'
0600 72 22                              jb      short BAD_NUMBER
0602 3C 39                              cmp     al, '9'
0604 77 1E                              ja      short BAD_NUMBER
0606 2C 30                              sub     al, '0'
0608 98                                 cbw
0609 03 C8                              add     cx, ax
060B 26: FE 0E 01C0 R                   dec     es:CURSOR_LOCATION_OR_SOMETHING
0610 26: 80 3E 01C0 R 00                cmp     es:CURSOR_LOCATION_OR_SOMETHING, 0
0616 74 0A                              je      short LAST_DIGIT
0618 B8 000A                            mov     ax, 10
061B F7 E1                              mul     cx
061D 8B C8                              mov     cx, ax
061F 43                                 inc     bx
0620 EB D9                              jmp     short NEXT_DIGIT
0622                    LAST_DIGIT:
0622 F8                                 clc
0623 C3                                 ret
0624                    BAD_NUMBER:
0624 F9                                 stc
0625 C3                                 ret
0626                    PARSE_NUMBER            ENDP


0626                    VERIFY_PROGRESS_STH     PROC NEAR
0626 53                                 push    bx
0627 51                                 push    cx
0628 26: C7 06 01CE R 2020                 mov     es:VERIFY_PROGRESS_BUFFER, 2020h        ; Two spaces
062F 26: C7 06 01D0 R 2020                 mov     es:VERIFY_PROGRESS_BUFFER_0, 2020h      ; Two spaces
0636 26: C7 06 01D2 R 2020                 mov     es:VERIFY_PROGRESS_BUFFER_1, 2020h      ; Two spaces
063D BB 0004                            mov     bx, 4
0640 B9 000A                            mov     cx, 0Ah
0643 26: A1 01D6 R                      mov     ax, word ptr es:VERIFY_STH_2
0647                    ANOTHER_PROGRESS_DIGIT:
0647 33 D2                              xor     dx, dx
0649 F7 F1                              div     cx
064B 80 C2 30                           add     dl, '0'
064E 26: 88 97 01CE R                   mov     byte ptr es:VERIFY_PROGRESS_BUFFER[bx], dl
0653 4B                                 dec     bx
0654 3D 0000                            cmp     ax, 0
0657 75 EE                              jne     short ANOTHER_PROGRESS_DIGIT
0659 59                                 pop     cx
065A 5B                                 pop     bx
065B C3                                 ret
065C                    VERIFY_PROGRESS_STH     ENDP


065C                    BACKSPACE_WIPE_WITH_SPACE       PROC NEAR
065C 51                                 push    cx
065D E8 076C R                          call    LOAD_CURSOR_POS
0660 50                                 push    ax
0661 E8 06D3 R                          call    READ_CURSOR_POS
0664 BA 099E R                          mov     dx, offset STRING_SPACE
0667 E8 074C R                          call    PRINT_STRING_NO_CRLF
066A E8 06EE R                          call    SET_CURSOR_POS
066D 58                                 pop     ax
066E E8 0776 R                          call    STORE_CURSOR_POS
0671 59                                 pop     cx
0672 C3                                 ret
0673                    BACKSPACE_WIPE_WITH_SPACE       ENDP


0673                    UNK_UNUSED_1            PROC NEAR
0673 26: A1 01C8 R                      mov     ax, es:LAST_CYLINDER_NUMBER
0677 33 C9                              xor     cx, cx
0679 26: 8A 0E 01CC R                   mov     cl, es:LAST_HEAD_NUMBER
067E F7 E1                              mul     cx
0680 B9 012C                            mov     cx, 300
0683 F7 F1                              div     cx

0685 3C 00                              cmp     al, 0
0687 77 02                              ja      short STILL_MORE
0689 FE C0                              inc     al
068B                    STILL_MORE:

068B 26: A2 01D4 R                      mov     es:UNK_UNUSED_STH, al
068F F8                                 clc
0690 C3                                 ret
0691                    UNK_UNUSED_1            ENDP


0691                    UNUSED_READ_CONTROL_BYTE        PROC NEAR
0691 BA 09C9 R                          mov     dx, offset STRING_CONTROL_BYTE_2
0694 E8 0755 R                          call    PRINT_CRLF_STRING
0697 E8 06D3 R                          call    READ_CURSOR_POS
069A 26: FE 0E 00B0 R                   dec     es:SAVED_COLUMN
069F E8 06EE R                          call    SET_CURSOR_POS
06A2 26: C6 06 01BB R 03                mov     es:READ_NUMBER_MAX_DIGITS, 3
06A8 26: C6 06 01BA R 03                mov     es:READ_NUMBER_MIN_DIGITS, 3
06AE E8 0527 R                          call    READ_NUMBER
06B1 26: A1 00B8 R                      mov     ax, es:READ_NUMBER_RESULT
06B5 26: 80 3E 00BA R 00                cmp     es:READ_NUMBER_RESULT_LEN, 0
06BB 75 02                              jne     short CONTROL_BYTE_SET
06BD B0 02                              mov     al, 2                   ; Default if unset
06BF                    CONTROL_BYTE_SET:
06BF 26: A2 01D5 R                      mov     es:CONTROL_BYTE, al
06C3 F8                                 clc
06C4 C3                                 ret
06C5                    UNUSED_READ_CONTROL_BYTE        ENDP


```

```
                        PAGE
06C5                    INT13H_FOR_CHOSEN_DRIVE PROC NEAR
06C5 26: 8A 16 01B9 R           mov     dl, es:HARD_DISK_NUMBER
06CA 80 CA 80                   or      dl, 80h
06CD B9 0001                    mov     cx, 1
06D0 CD 13                      int     13h
06D2 C3                         ret
06D3                    INT13H_FOR_CHOSEN_DRIVE ENDP


06D3                    READ_CURSOR_POS         PROC NEAR
06D3 50                         push    ax
06D4 56                         push    si
06D5 53                         push    bx
06D6 52                         push    dx
06D7 51                         push    cx
06D8 B4 03                      mov     ah, 3                   ; Read video cursor
06DA B7 00                      mov     bh, 0
06DC CD 10                      int     10h
06DE 26: 88 36 00B1 R           mov     es:SAVED_ROW, dh
06E3 26: 88 16 00B0 R           mov     es:SAVED_COLUMN, dl
06E8 59                         pop     cx
06E9 5A                         pop     dx
06EA 5B                         pop     bx
06EB 5E                         pop     si
06EC 58                         pop     ax
06ED C3                         ret
06EE                    READ_CURSOR_POS         ENDP


06EE                    SET_CURSOR_POS          PROC NEAR
06EE 50                         push    ax
06EF 56                         push    si
06F0 52                         push    dx
06F1 53                         push    bx
06F2 B4 02                      mov     ah, 2                   ; Set video cursor
06F4 26: 8A 36 00B1 R           mov     dh, es:SAVED_ROW
06F9 26: 8A 16 00B0 R           mov     dl, es:SAVED_COLUMN
06FE B7 00                      mov     bh, 0
0700 CD 10                      int     10h
0702 5B                         pop     bx
0703 5A                         pop     dx
0704 5E                         pop     si
0705 58                         pop     ax
0706 C3                         ret
0707                    SET_CURSOR_POS          ENDP


0707                    PRINT_MESSAGE_AND_DUMP_CONTROLLER_BUFFER        PROC NEAR
0707 E8 0755 R                  call    PRINT_CRLF_STRING
070A 1E                         push    ds
070B E8 1139 R                  call    ZERO_DS
                                ASSUME  ds:ZEROSEG
070E A0 0442 R                  mov     al, BDA_CONTROLLER_DATA_BUFFER_00
0711 E8 072E R                  call    PRINT_HEX_BYTE
0714 A0 0443 R                  mov     al, BDA_CONTROLLER_DATA_BUFFER_01
0717 E8 072E R                  call    PRINT_HEX_BYTE
071A A0 0444 R                  mov     al, BDA_CONTROLLER_DATA_BUFFER_02
071D E8 072E R                  call    PRINT_HEX_BYTE
0720 A0 0445 R                  mov     al, BDA_CONTROLLER_DATA_BUFFER_03
0723 E8 072E R                  call    PRINT_HEX_BYTE
0726 BA 09D9 R                  mov     dx, offset STRING_CRLF_DOS
0729 E8 074C R                  call    PRINT_STRING_NO_CRLF
072C 1F                         pop     ds
072D C3                         ret
072E                    PRINT_MESSAGE_AND_DUMP_CONTROLLER_BUFFER        ENDP


072E                    PRINT_HEX_BYTE          PROC NEAR
072E 50                         push    ax
072F 32 FF                      xor     bh, bh
0731 B0 20                      mov     al, ' '
0733 E8 07AB R                  call    BIOS_WRITE_TEXT
0736 58                         pop     ax
0737 50                         push    ax
0738 B1 04                      mov     cl, 4
073A D2 E8                      shr     al, cl
073C E8 07B0 R                  call    NUMBER_TO_HEX_DIGIT
073F E8 07AB R                  call    BIOS_WRITE_TEXT
0742 58                         pop     ax
0743 24 0F                      and     al, 0Fh
0745 E8 07B0 R                  call    NUMBER_TO_HEX_DIGIT
0748 E8 07AB R                  call    BIOS_WRITE_TEXT
074B C3                         ret
074C                    PRINT_HEX_BYTE          ENDP


074C                    PRINT_STRING_NO_CRLF    PROC NEAR
074C 1E                         push    ds
074D 0E                         push    cs
074E 1F                         pop     ds
                                ASSUME  ds:ROM
074F B4 09                      mov     ah, 9           ; DOS print string
0751 CD 21                      int     21h
0753 1F                         pop     ds
                                ASSUME  ds:FORMATTER_HEAP
0754 C3                         ret
0755                    PRINT_STRING_NO_CRLF    ENDP


```

```
                                PAGE
0755                            PRINT_CRLF_STRING       PROC NEAR
0755 1E                                 push    ds
0756 0E                                 push    cs
0757 1F                                 pop     ds
                                        ASSUME  ds:ROM
0758 B4 09                              mov     ah, 9           ; DOS print string
075A 52                                 push    dx
075B BA 09D9 R                          mov     dx, offset STRING_CRLF_DOS
075E CD 21                              int     21h
0760 5A                                 pop     dx
0761 CD 21                              int     21h
0763 1F                                 pop     ds
                                        ASSUME  ds:FORMATTER_HEAP
0764 C3                                 ret
0765                            PRINT_CRLF_STRING       ENDP


0765                            DO_BEEP                 PROC NEAR
0765 BA 094F R                          mov     dx, offset STRING_BEEP
0768 E8 074C R                          call    PRINT_STRING_NO_CRLF
076B C3                                 ret
076C                            DO_BEEP                 ENDP


076C                            LOAD_CURSOR_POS         PROC NEAR
076C 26: A0 00B0 R                      mov     al, es:SAVED_COLUMN
0770 26: 8A 26 00B1 R                   mov     ah, es:SAVED_ROW
0775 C3                                 ret
0776                            LOAD_CURSOR_POS         ENDP


0776                            STORE_CURSOR_POS        PROC NEAR
0776 26: A2 00B0 R                      mov     es:SAVED_COLUMN, al
077A 26: 88 26 00B1 R                   mov     es:SAVED_ROW, ah
077F C3                                 ret
0780                            STORE_CURSOR_POS        ENDP


0780                            BAD_ENTRY_BEEP          PROC NEAR
0780 E8 06D3 R                          call    READ_CURSOR_POS
0783 E8 0765 R                          call    DO_BEEP
0786 26: FE 0E 00B1 R                   dec     es:SAVED_ROW
078B 26: C6 06 00B0 R 00                mov     es:SAVED_COLUMN, 0
0791 E8 06EE R                          call    SET_CURSOR_POS
0794 C3                                 ret
0795                            BAD_ENTRY_BEEP          ENDP


0795                            MOVE_CURSOR             PROC NEAR
0795 E8 06D3 R                          call    READ_CURSOR_POS
0798 26: A0 00B0 R                      mov     al, es:SAVED_COLUMN
079C 26: 8A 26 00B1 R                   mov     ah, es:SAVED_ROW
07A1 2C 05                              sub     al, 5
07A3 26: A2 00B0 R                      mov     es:SAVED_COLUMN, al
07A7 E8 06EE R                          call    SET_CURSOR_POS
07AA C3                                 ret
07AB                            MOVE_CURSOR             ENDP


07AB                            BIOS_WRITE_TEXT         PROC NEAR
07AB B4 0E                              mov     ah, 0Eh         ; Write string
07AD CD 10                              int     10h
07AF C3                                 ret
07B0                            BIOS_WRITE_TEXT         ENDP


07B0                            NUMBER_TO_HEX_DIGIT     PROC NEAR
07B0 3C 0A                              cmp     al, 10
07B2 72 05                              jb      short ZERO_TO_NINE
07B4 2C 0A                              sub     al, 10
07B6 04 41                              add     al, 'A'
07B8 C3                                 ret
07B9                            ZERO_TO_NINE:
07B9 04 30                              add     al, '0'
07BB C3                                 ret
07BC                            NUMBER_TO_HEX_DIGIT     ENDP


07BC 54 68 69 73 20 46  STRING_FORMAT_WILL_DESTROY      DB              "This FORMAT routine will DESTROY"
     4F 52 4D 41 54 20
     72 6F 75 74 69 6E
     65 20 77 69 6C 6C
     20 44 45 53 54 52
     4F 59
07DC 20 41 4C 4C 20 64                                  DB              " ALL data on your disk!$"
     61 74 61 20 6F 6E
     20 79 6F 75 72 20
     64 69 73 6B 21 24
07F4 0A 45 6E 74 65 72  STRING_ENTER_DRIVE               DB      0Ah,    "Enter drive # (0 or 1):$"
     20 64 72 69 76 65
     20 23 20 28 30 20
     6F 72 20 31 29 3A
     24
080D 07 46 6F 72 6D 61  STRING_FORMAT_FAILED            DB          7, "Format failed$"
     74 20 66 61 69 6C
     65 64 24
081C 0A 07 46 6C 61 67  STRING_BAD_SECTOR_FAILED        DB      0Ah, 7, "Flag bad track failed$"
     20 62 61 64 20 74
     72 61 63 6B 20 66
     61 69 6C 65 64 24
0834 0A 07 56 65 72 69  STRING_VERIFY_FAILED            DB      0Ah, 7, "Verify failed$"
     66 79 20 66 61 69
     6C 65 64 24
0844 07 49 6E 69 74 20  UNUSED_STRING_INIT_FAILURE      DB          7, "Init failure - aborted$"
     66 61 69 6C 75 72
     65 20 2D 20 61 62
     6F 72 74 65 64 24
085C 0A 46 6F 72 6D 61  STRING_FORMATTING               DB      0Ah,    "Formatting...$"
     74 74 69 6E 67 2E
     2E 2E 24
086B 46 6F 72 6D 61 74  STRING_FORMAT_COMPLETE          DB              "Format complete$"
     20 63 6F 6D 70 6C
```

```
1344      65 74 65 24
1345 087B 56 65 72 69 66 79     STRING_VERIFYING                DB              "Verifying...  $"
1346      69 6E 67 2E 2E 2E
1347      20 20 24
1348 088A 0A 56 65 72 69 66     STRING_VERIFY_COMPLETE          DB      0Ah,    "Verify complete$"
1349      79 20 63 6F 6D 70
1350      6C 65 74 65 24
1351 089B 4E 6F 20 6D 6F 72     STRING_NO_MORE_DEFECTS          DB              "No more defects accepted!$"
1352      65 20 64 65 66 65
1353      63 74 73 20 61 63
1354      63 65 70 74 65 64
1355      21 24
1356 08B5 0A 41 6E 79 20 64     STRING_ANY_DEFECTS              DB      0Ah,    "Any defects (Y/N)? $"
1357      65 66 65 63 74 73
1358      20 28 59 2F 4E 29
1359      3F 20 24
1360 08CA 0A 28 50 72 65 73     STRING_PRESS_RET_TO_END         DB      0Ah,    "(Press <RET> to end defect list)$"
1361      73 20 3C 52 45 54
1362      3E 20 74 6F 20 65
1363      6E 64 20 64 65 66
1364      65 63 74 20 6C 69
1365      73 74 29 24
1366 08EC 43 59 4C 49 4E 44     STRING_CYLINDER                 DB              "CYLINDER:     $"
1367      45 52 3A 20 20 20
1368      20 20 24
1369 08FB 20 20 20 20 48 45     STRING_HEAD                     DB              "    HEAD:     $"
1370      41 44 3A 20 20 20
1371      20 20 24
1372 090A 4D 6F 72 65 20 65     STRING_ASK_MORE_DEFECT_ENTRIES  DB              "More entries (Y/N)? $"
1373      6E 74 72 69 65 73
1374      20 28 59 2F 4E 29
1375      3F 20 24
1376 091F 0A 41 72 65 20 79     STRING_CONFIRM_FORMAT           DB      0Ah,    "Are you SURE you want to format (Y/N)? $"
1377      6F 75 20 53 55 52
1378      45 20 79 6F 75 20
1379      77 61 6E 74 20 74
1380      6F 20 66 6F 72 6D
1381      61 74 20 28 59 2F
1382      4E 29 3F 20 24
1383 0948 20 20 20 20 20 20     STRING_SPACES                   DB              "      $"
1384      24
1385 094F 07 24                 STRING_BEEP                     DB          7, "$"
1386 0951 DB                                                    DB      0DBh
1387 0952 6C                                                    DB      6Ch
1388 0953 0A 50 72 65 73 73     STRING_PRESS_RET_TO_PROCEED     DB      0Ah,    "Press <RET> to proceed or <ESC> to cancel...$"
1389      20 3C 52 45 54 3E
1390      20 74 6F 20 70 72
1391      6F 63 65 65 64 20
1392      6F 72 20 3C 45 53
1393      43 3E 20 74 6F 20
1394      63 61 6E 63 65 6C
1395      2E 2E 2E 24
1396 0981 50 72 65 73 73 20     STRING_ANY_KEY_TO_REBOOT        DB              "Press any key to reboot...$"
1397      61 6E 79 20 6B 65
1398      79 20 74 6F 20 72
1399      65 62 6F 6F 74 2E
1400      2E 2E 24
1401 099C 25 24                 STRING_PERCENT                  DB              "%$"
1402 099E 20 24                 STRING_SPACE                    DB              " $"
1403 09A0 07 57 72 69 74 65     STRING_BEEP_SECTOR_WRITE_FAIL   DB          7, "Write buffer failed$"
1404      20 62 75 66 66 65
1405      72 20 66 61 69 6C
1406      65 64 24
1407 09B5 0A 49 6E 74 65 72     STRING_INTERLEAVE               DB      0Ah,    "Interleave (1-15):$"
1408      6C 65 61 76 65 20
1409      28 31 2D 31 35 29
1410      3A 24
1411 09C9 43 4F 4E 54 52 4F     STRING_CONTROL_BYTE_2           DB              "CONTROL BYTE: 2$"
1412      4C 20 42 59 54 45
1413      3A 20 32 24
1414 09D9 0D 0A 24              STRING_CRLF_DOS                 DB      0Dh,0Ah,"$"
1415 09DC 00 08 10 00 08 00                                     DB      0, 8, 10h, 0, 8, 0, 8, 7, 2, 4, 4, 4, 0, 0, 0, 0
1416      08 07 02 04 04 04
1417      00 00 00 00
1418 09EC 50 72 6F 63 65 73     STRING_PROCESSING_DEFECTS       DB              "Processing defects...$"
1419      73 69 6E 67 20 64
1420      65 66 65 63 74 73
1421      2E 2E 2E 24
1422
1423 0A02 28 63 29 43 6F 70     COPYRIGHT_STRING                DB      "(c)Copyright 1987 SMS"
1424      79 72 69 67 68 74
1425      20 31 39 38 37 20
1426      53 4D 53
1427                                                                    ; Yeah, do not disassemble or anything.
1428                                                                    ; You'll be shot in the face if you do.
1429
1430
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Option ROM initialization                       :
                        ;--------------------------------------------------------

0A17                    ROM_INIT                PROC FAR
0A17 E8 1141 R                  call    ZERO_DS_ALT
                                ASSUME  ds:ZEROSEG

0A1A FA                         cli
0A1B C7 06 0064 R 1148 R        mov     INT19H_OFFSET, offset INT19H_HANDLER
0A21 8C 0E 0066 R               mov     INT19H_SEGMENT, cs
0A25 C7 06 0104 R 0020 R        mov     INT41H_OFFSET, offset INT41H_DATA
0A2B 8C 0E 0106 R               mov     INT41H_SEGMENT, cs
0A2F 06                         push    es
0A30 57                         push    di
0A31 C4 3E 004C R               les     di, dword ptr INT13H_OFFSET
0A35 89 3E 0100 R               mov     INT40H_OFFSET, di
0A39 8C 06 0102 R               mov     INT40H_SEGMENT, es
                                ASSUME  es:ZEROSEG
0A3D 5F                         pop     di
0A3E 07                         pop     es
0A3F C7 06 004C R 0B77 R        mov     INT13H_OFFSET, offset INT13H_HANDLER
0A45 8C 0E 004E R               mov     INT13H_SEGMENT, cs
0A49 FB                         sti

0A4A E8 1139 R                  call    ZERO_DS
                                ASSUME  ds:ZEROSEG
0A4D C6 06 0475 R 00            mov     BDA_NUMBER_OF_HARD_DISKS, 0

0A52 FA                         cli
0A53 E4 21                      in      al, 21h
0A55 24 FE                      and     al, 0FEh                        ; Unmask IRQ0 (timer)
0A57 E6 21                      out     21h, al
0A59 FB                         sti

0A5A E8 0D61 R                  call    MASTER_RESET
0A5D B4 14                      mov     ah, INT13H_OP_14_CONTROLLER_DIAGS
0A5F E8 0B5A R                  call    D0_CALL_INT13H
0A62 73 06                      jnc     short CONTROLLER_DIAGS_FINE
0A64 BE 1216 R                  mov     si, offset STRING_B
0A67 E9 0B00 R                  jmp     DISK_INIT_ERROR
0A6A                    CONTROLLER_DIAGS_FINE:
0A6A B4 12                      mov     ah, INT13H_OP_12_SRAM_DIAGS
0A6C E8 0B5A R                  call    D0_CALL_INT13H
0A6F 73 06                      jnc     short SRAM_DIAGS_FINE
0A71 BE 1214 R                  mov     si, offset STRING_A
0A74 E9 0B00 R                  jmp     DISK_INIT_ERROR
0A77                    SRAM_DIAGS_FINE:
0A77 50                         push    ax
0A78 53                         push    bx
0A79 51                         push    cx
0A7A 52                         push    dx
0A7B B8 01C2                    mov     ax, 450                         ; Drive ready (spin-up) timeout
0A7E 81 3E 0472 R 04D2          cmp     BDA_SOFT_RESET_FLAG, 4D2h       ; If the soft reset flag is not present
0A84 75 03                      jne     short NOT_SOFT_RESET_4D2        ;  we keep the long timeout of 450,
0A86 B8 005A                    mov     ax, 90                          ;  otherwise we shorten it to 90
0A89                    NOT_SOFT_RESET_4D2:
0A89 E8 1028 R                  call    BDA_TIMER_TO_CX_DX
0A8C 03 D0                      add     dx, ax
0A8E 83 D1 00                   adc     cx, 0
0A91 8B D9                      mov     bx, cx
0A93 8B C2                      mov     ax, dx

0A95                    D0_TEST_READY:
0A95 50                         push    ax
0A96 53                         push    bx
0A97 B4 10                      mov     ah, INT13H_OP_10_TEST_DRIVE_READY
0A99 E8 0B5A R                  call    D0_CALL_INT13H
0A9C 73 1E                      jnc     short D0_RECALIBRATE
0A9E 80 FC 27                   cmp     ah, INT13H_STATUS_27_NEED_RECALIBRATE
0AA1 74 19                      je      short D0_RECALIBRATE
0AA3 E8 1028 R                  call    BDA_TIMER_TO_CX_DX
0AA6 5B                         pop     bx
0AA7 58                         pop     ax
0AA8 3B D9                      cmp     bx, cx
0AAA 77 E9                      ja      short D0_TEST_READY
0AAC 72 04                      jb      short D0_TEST_READY_TIMED_OUT
0AAE 3B C2                      cmp     ax, dx
0AB0 77 E3                      ja      short D0_TEST_READY

0AB2                    D0_TEST_READY_TIMED_OUT:
0AB2 5A                         pop     dx
0AB3 59                         pop     cx
0AB4 5B                         pop     bx
0AB5 58                         pop     ax
0AB6 BE 1218 R                  mov     si, offset STRING_C
0AB9 EB 45                      jmp     short DISK_INIT_ERROR
0ABB 90                         nop

0ABC                    D0_RECALIBRATE:
0ABC 5B                         pop     bx
0ABD 58                         pop     ax
0ABE 5A                         pop     dx
0ABF 59                         pop     cx
0AC0 5B                         pop     bx
0AC1 58                         pop     ax
0AC2 B4 11                      mov     ah, INT13H_OP_11_RECALIBRATE
0AC4 E8 0B5A R                  call    D0_CALL_INT13H
0AC7 73 06                      jnc     short D0_RESET_DISK_SYSTEM
0AC9 BE 121A R                  mov     si, offset STRING_D
0ACC EB 32                      jmp     short DISK_INIT_ERROR
0ACE 90                         nop

0ACF                    D0_RESET_DISK_SYSTEM:
0ACF B4 00                      mov     ah, INT13H_OP_00_RESET_DISK_SYSTEM
0AD1 E8 0B5A R                  call    D0_CALL_INT13H
0AD4 73 06                      jnc     short D0_RESET_SUCCESSFUL
0AD6 BE 121C R                  mov     si, offset STRING_E
0AD9 E8 0B00 R                  call    DISK_INIT_ERROR
```

```
0ADC                    D0_RESET_SUCCESSFUL:
0ADC FE 06 0475 R                       inc     BDA_NUMBER_OF_HARD_DISKS

0AE0 B4 10                              mov     ah, INT13H_OP_10_TEST_DRIVE_READY
0AE2 E8 0B68 R                          call    D1_CALL_INT13H
0AE5 73 08                              jnc     short D1_RECALIBRATE
0AE7 80 FC 27                           cmp     ah, INT13H_STATUS_27_NEED_RECALIBRATE
0AEA 74 03                              je      short D1_RECALIBRATE
0AEC EB 15                              jmp     short ROM_INIT_DONE
0AEE 90                                 nop

0AEF                    D1_RECALIBRATE:
0AEF B4 11                              mov     ah, INT13H_OP_11_RECALIBRATE
0AF1 E8 0B68 R                          call    D1_CALL_INT13H
0AF4 73 03                              jnc     short D1_RECALIBRATE_SUCCESSFUL
0AF6 EB 0B                              jmp     short ROM_INIT_DONE
0AF8 90                                 nop

0AF9                    D1_RECALIBRATE_SUCCESSFUL:
0AF9 FE 06 0475 R                       inc     BDA_NUMBER_OF_HARD_DISKS
0AFD EB 04                              jmp     short ROM_INIT_DONE
0AFF 90                                 nop

0B00                    DISK_INIT_ERROR:
0B00 E8 0B16 R                          call    ERROR_1701
0B03                    ROM_INIT_DONE:
0B03 E8 1139 R                          call    ZERO_DS
                                        ASSUME  ds:ZEROSEG
0B06 32 C0                              xor     al, al                          ; Disable DMA/IRQ
0B08 BA 0323                            mov     dx, IO_PORT_323_DMA_IRQ
0B0B E8 103A R                          call    DO_NOTHING
0B0E E8 1137 R                          call    OUTB_DX_AL
0B11 B0 07                              mov     al, 111b                        ; Mask DRQ3
0B13 E6 0A                              out     0Ah, al
0B15 CB                                 ret
0B16                    ROM_INIT                ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       This prints the "1701-?" error code when        :
                        ;       the option ROM fails to bring up the drive.     :
                        ;                                                       :
                        ;       The SI register points to -A, -B, ... string    :
                        ;       which details the error encountered.            :
                        ;--------------------------------------------------------

0B16                    ERROR_1701              PROC NEAR
0B16 1E                         push    ds
0B17 56                         push    si
0B18 BD 000F                    mov     bp, 0Fh
0B1B 32 FF                      xor     bh, bh
0B1D BE 1210 R                  mov     si, offset STRING_1701
0B20 B9 0004                    mov     cx, 4
0B23 90                         nop
0B24 0E                         push    cs
0B25 1F                         pop     ds
                                ASSUME  ds:ROM
0B26 E8 0B3C R                  call    PRINT_CX_CHARS_FROM_DS_SI ; Print "1701"
0B29 5E                         pop     si
0B2A B9 0002                    mov     cx, 2
0B2D E8 0B3C R                  call    PRINT_CX_CHARS_FROM_DS_SI ; Print (-A, -B, -C, -D, ...)
0B30 BE 120E R                  mov     si, offset STRING_CRLF
0B33 B9 0002                    mov     cx, 2
0B36 90                         nop
0B37 E8 0B3C R                  call    PRINT_CX_CHARS_FROM_DS_SI ; Print "\r\n"
0B3A 1F                         pop     ds
0B3B C3                         ret
0B3C                    ERROR_1701              ENDP


                        ;--------------------------------------------------------
                        ;       A string print routine.                         :
                        ;--------------------------------------------------------

0B3C                    PRINT_CX_CHARS_FROM_DS_SI       PROC NEAR
0B3C 50                         push    ax
0B3D 56                         push    si
0B3E                    NEXT_CHAR:
0B3E AC                         lodsb
0B3F B4 0E                      mov     ah, 0Eh
0B41 CD 10                      int     10h                     ; TTY write
0B43 49                         dec     cx
0B44 75 F8                      jne     short NEXT_CHAR
0B46 5E                         pop     si
0B47 58                         pop     ax
0B48 C3                         ret
0B49                    PRINT_CX_CHARS_FROM_DS_SI       ENDP


                        ;--------------------------------------------------------
                        ;       Another string print routine. Unused.           :
                        ;--------------------------------------------------------

0B49                    UNUSED_PRINT_CX_CHARS_FROM_CS_SI        PROC NEAR
0B49 50                         push    ax
0B4A 56                         push    si
0B4B 1E                         push    ds
0B4C 0E                         push    cs
0B4D 1F                         pop     ds
                                ASSUME  ds:ROM
0B4E                    UNUSED_NEXT_CHAR:
0B4E AC                         lodsb
0B4F B4 0E                      mov     ah, 0Eh                 ; TTY write
0B51 CD 10                      int     10h
0B53 49                         dec     cx
0B54 75 F8                      jne     short UNUSED_NEXT_CHAR
0B56 1F                         pop     ds
                                ASSUME  ds:ZEROSEG
0B57 5E                         pop     si
0B58 58                         pop     ax
0B59 C3                         ret
0B5A                    UNUSED_PRINT_CX_CHARS_FROM_CS_SI        ENDP


                        ;--------------------------------------------------------
                        ;       Issue a disk service routine with some          :
                        ;       pre-set arguments for the first drive.          :
                        ;--------------------------------------------------------

0B5A                    D0_CALL_INT13H          PROC NEAR
0B5A 33 D2                      xor     dx, dx
0B5C 81 CA 0080                 or      dx, 80h         ; DL = 80H = Drive 0
0B60 B9 0001                    mov     cx, 1           ; CH=0 = First Cylinder, CL=1 = First Sector
0B63 B0 00                      mov     al, 0           ; AL=0 = Number of Sectors
0B65 CD 13                      int     13h
0B67 C3                         ret
0B68                    D0_CALL_INT13H          ENDP


                        ;--------------------------------------------------------
                        ;       Issue a disk service routine with some          :
                        ;       pre-set arguments for the second drive.         :
                        ;--------------------------------------------------------

0B68                    D1_CALL_INT13H          PROC NEAR
0B68 BA 0001                    mov     dx, 1
0B6B 81 CA 0080                 or      dx, 80h         ; DL = 81H = Drive 1
0B6F B9 0001                    mov     cx, 1           ; CH=0 = First Cylinder, CL=1 = First Sector
0B72 B0 00                      mov     al, 0           ; AL=0 = Number of Sectors
0B74 CD 13                      int     13h
0B76 C3                         ret
0B77                    D1_CALL_INT13H          ENDP


```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       BIOS Disk Services (INT 13H)                    :
                        ;--------------------------------------------------------

0B77                    INT13H_HANDLER          PROC FAR
0B77 FB                                 sti
0B78 F6 C2 80                           test    dl, 80h
0B7B 74 02                              je      short NOT_A_HARD_DISK
0B7D EB 05                              jmp     short NOT_A_FLOPPY
0B7F                    NOT_A_HARD_DISK:
0B7F CD 40                              int     40h                     ; Invoke the old (floppy)
0B81 EB 75                              jmp     short ALL_DONE          ;  INT13H handler
0B83 90                                 nop
0B84                    NOT_A_FLOPPY:
0B84 F6 C4 FF                           test    ah, 0FFh                ; Test for AH=00H ("Disk Subsystem Reset")
0B87 75 04                              jne     short HARD_DISK         ; Handle a hard disk function
0B89 50                                 push    ax
0B8A CD 40                              int     40h                     ; Invoke the floppy reset handler and
                                                                        ;  then also proceed with the hard disk
0B8C 58                                 pop     ax
0B8D                    HARD_DISK:
0B8D 56                                 push    si
0B8E 51                                 push    cx
0B8F BE 121E R                          mov     si, offset INT13H_TRIVIAL_OP_VECTOR
0B92 B9 0002                            mov     cx, 2                           ; Number of vector entries
0B95                    NEXT_TRIVIAL_OP:
0B95 2E: 3A 24                          cmp     ah, cs:[si]
0B98 74 0A                              je      short TRIVIAL_OP_FOUND
0B9A 83 C6 03                           add     si, 3                           ; Size of vector entry
0B9D E2 F6                              loop    NEXT_TRIVIAL_OP
0B9F 59                                 pop     cx
0BA0 5E                                 pop     si
0BA1 EB 06                              jmp     short NO_TRIVIAL_OP_FOUND
0BA3 90                                 nop
0BA4                    TRIVIAL_OP_FOUND:
0BA4 59                                 pop     cx
0BA5 2E: FF 64 01                       jmp     word ptr cs:[si+1]
0BA9                    NO_TRIVIAL_OP_FOUND:
0BA9 57                                 push    di
0BAA 56                                 push    si
0BAB 06                                 push    es
0BAC 1E                                 push    ds
0BAD 55                                 push    bp
0BAE 52                                 push    dx
0BAF 51                                 push    cx
0BB0 53                                 push    bx
0BB1 80 FC 00                           cmp     ah, INT13H_OP_00_RESET_DISK_SYSTEM
0BB4 75 02                              jne     short PROCEED_HANDLING_FUNC
0BB6 B2 80                              mov     dl, 80h                         ; Floppy reset has been done.
                                                                                ;  Reset the hard disk now.
0BB8                    PROCEED_HANDLING_FUNC:
0BB8 E8 1139 R                          call    ZERO_DS
                                        ASSUME  ds:ZEROSEG
0BBB E8 0DFC R                          call    CHECK_VALID_DISK_NUMBER
0BBE 73 08                              jnb     short DISK_NUMBER_IS_VALID
0BC0 C6 06 0474 R 01                    mov     BDA_LAST_OP_STATUS, INT13H_STATUS_01_BAD_COMMAND
0BC5 EB 0C                              jmp     short CLEANUP
0BC7 90                                 nop
0BC8                    DISK_NUMBER_IS_VALID:
0BC8 E8 0BFB R                          call    SET_BDA_STATUS_SUCCESS
0BCB E8 0C06 R                          call    FILL_COMMAND_BUFFER
0BCE 72 03                              jc      short CLEANUP
0BD0 E8 0CB0 R                          call    CONTROLLER_FUNC
0BD3                    CLEANUP:
0BD3 50                                 push    ax
0BD4 E8 1139 R                          call    ZERO_DS
                                        ASSUME  ds:ZEROSEG
0BD7 32 C0                              xor     al, al                          ; Disable DMA/IRQ
0BD9 BA 0323                            mov     dx, IO_PORT_323_DMA_IRQ
0BDC E8 103A R                          call    DO_NOTHING
0BDF E8 1137 R                          call    OUTB_DX_AL
0BE2 B0 07                              mov     al, 111b                        ; Mask DRQ3
0BE4 E6 0A                              out     0Ah, al
0BE6 58                                 pop     ax
0BE7 8A 26 0474 R                       mov     ah, BDA_LAST_OP_STATUS
0BEB 0A E4                              or      ah, ah
0BED 74 01                              je      short NO_ERROR
0BEF F9                                 stc
0BF0                    NO_ERROR:
0BF0 5B                                 pop     bx
0BF1 59                                 pop     cx
0BF2 5A                                 pop     dx
0BF3 5D                                 pop     bp
0BF4 1F                                 pop     ds
0BF5 07                                 pop     es
0BF6 5E                                 pop     si
0BF7 5F                                 pop     di
0BF8                    ALL_DONE:
0BF8 CA 0002                            ret     2
0BFB                    INT13H_HANDLER          ENDP


                        ;--------------------------------------------------------
                        ;       Reset the last disk operation status            :
                        ;--------------------------------------------------------

0BFB                    SET_BDA_STATUS_SUCCESS  PROC NEAR
0BFB 1E                                 push    ds
0BFC E8 1139 R                          call    ZERO_DS
                                        ASSUME  ds:ZEROSEG
0BFF C6 06 0474 R 00                    mov     BDA_LAST_OP_STATUS, INT13H_STATUS_00_NO_ERROR
0C04 1F                                 pop     ds
0C05 C3                                 ret
0C06                    SET_BDA_STATUS_SUCCESS  ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       This takes the INT13H arguments from            :
                        ;       registers and places them in command buffer     :
                        ;       for submission to the controller.               :
                        ;--------------------------------------------------------

0C06                    FILL_COMMAND_BUFFER     PROC NEAR
0C06 E8 0C7E R                  call    VALIDATE_CHS
0C09 73 03                      jnb     short CHS_VALID
0C0B E9 0CD1 R                  jmp     BAD_COMMAND_ERROR_RETURN
0C0E                    CHS_VALID:
0C0E 50                         push    ax
0C0F 53                         push    bx
0C10 8A C4                      mov     al, ah
0C12 BB 122A R                  mov     bx, offset INT13H_OP_TO_COMMAND_BYTE
0C15 2E: D7                     xlat    byte ptr cs:[bx]
0C17 A2 0442 R                  mov     BDA_CONTROLLER_DATA_BUFFER_00, al
0C1A 5B                         pop     bx
0C1B 58                         pop     ax
0C1C FE C9                      dec     cl
0C1E 89 0E 0444 R               mov     word ptr BDA_CONTROLLER_DATA_BUFFER_02, cx
0C22 A2 0446 R                  mov     BDA_CONTROLLER_DATA_BUFFER_04, al
0C25 80 FC 0E                   cmp     ah, OP_0E_READ_SECTOR_BUFFER
0C28 74 05                      je      short READ_OR_WRITE
0C2A 80 FC 0F                   cmp     ah, OP_0F_WRITE_SECTOR_BUFFER
0C2D 75 05                      jne     short ANY_OP
0C2F                    READ_OR_WRITE:
0C2F C6 06 0446 R 01            mov     BDA_CONTROLLER_DATA_BUFFER_04, 1
0C34                    ANY_OP:
0C34 80 E6 1F                   and     dh, 1Fh
0C37 88 36 0443 R               mov     BDA_CONTROLLER_DATA_BUFFER_01, dh
0C3B B1 03                      mov     cl, 3
0C3D D2 CA                      ror     dl, cl
0C3F 80 E2 60                   and     dl, 60h
0C42 08 16 0443 R               or      BDA_CONTROLLER_DATA_BUFFER_01, dl
0C46 50                         push    ax
0C47 57                         push    di
0C48 51                         push    cx
0C49 06                         push    es
0C4A E8 1139 R                  call    ZERO_DS
                                ASSUME  ds:ZEROSEG
0C4D E8 101E R                  call    DRIVE_PARAM_TO_DI
0C50 83 C7 08                   add     di, 8
0C53 26: 8A 05                  mov     al, es:[di]
0C56 A2 0447 R                  mov     BDA_CONTROLLER_DATA_BUFFER_05, al
0C59 80 3E 0442 R 04            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_04_FORMAT_DRIVE
0C5E 74 0E                      je      short FORMAT_DRIVE_OR_TRACK_NOT_BAD
0C60 80 3E 0442 R 06            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_06_FORMAT_TRACK
0C65 74 07                      je      short FORMAT_DRIVE_OR_TRACK_NOT_BAD
0C67 80 3E 0442 R 07            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_07_FORMAT_BAD_TRACK
0C6C 75 0A                      jne     short FORMAT_DRIVE_OR_TRACK
0C6E                    FORMAT_DRIVE_OR_TRACK_NOT_BAD:
0C6E 80 26 0444 R C0            and     BDA_CONTROLLER_DATA_BUFFER_02, 0C0h
0C73 80 0E 0447 R 00            or      BDA_CONTROLLER_DATA_BUFFER_05, 0
0C78                    FORMAT_DRIVE_OR_TRACK:
0C78 07                         pop     es
0C79 59                         pop     cx
0C7A 5F                         pop     di
0C7B 58                         pop     ax
0C7C F8                         clc
0C7D C3                         ret
0C7E                    FILL_COMMAND_BUFFER     ENDP


                        ;--------------------------------------------------------
                        ;       Checks that the coordinates are in bounds.      :
                        ;                                                       :
                        ;       Takes INT13H arguments in CX/DX,                :
                        ;       sets carry on failure.                          :
                        ;                                                       :
                        ;       Accepts C/H/S up to 614/4/17                    :
                        ;       Note that our drive actually is 615/2/34...     :
                        ;--------------------------------------------------------

0C7E                    VALIDATE_CHS            PROC NEAR
0C7E 50                         push    ax
0C7F 53                         push    bx
0C80 51                         push    cx
0C81 52                         push    dx
0C82 80 FA 01                   cmp     dl, 1           ; Check drive number
0C85 77 1F                      ja      short BAD_ARGUMENT
0C87 80 FE 03                   cmp     dh, 3           ; Check head number
0C8A 77 1A                      ja      short BAD_ARGUMENT
0C8C 51                         push    cx
0C8D 80 E1 3F                   and     cl, 3Fh
0C90 80 F9 11                   cmp     cl, 17          ; Check sector number
0C93 59                         pop     cx
0C94 77 10                      ja      short BAD_ARGUMENT
0C96 80 E1 C0                   and     cl, 0C0h
0C99 8B C1                      mov     ax, cx
0C9B B1 02                      mov     cl, 2
0C9D D2 C0                      rol     al, cl
0C9F 86 E0                      xchg    ah, al
0CA1 3D 0265                    cmp     ax, 613         ; Check cylinder
0CA4 76 04                      jbe     short ARGUMENTS_FINE
0CA6                    BAD_ARGUMENT:
0CA6 F9                         stc
0CA7 EB 02                      jmp     short DONE
0CA9 90                         nop
0CAA                    ARGUMENTS_FINE:
0CAA F8                         clc
0CAB                    DONE:
0CAB 5A                         pop     dx
0CAC 59                         pop     cx
0CAD 5B                         pop     bx
0CAE 58                         pop     ax
0CAF C3                         ret
0CB0                    VALIDATE_CHS            ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       This handles INT13H functions that require      :
                        ;       any sort of interaction with the controller     :
                        ;       hardware. This could be either a command        :
                        ;       submission, but also a reset.                   :
                        ;--------------------------------------------------------

0CB0                    CONTROLLER_FUNC         PROC NEAR
0CB0 80 FC 14                   cmp     ah, LAST_INT13H_OP
0CB3 77 1C                      ja      short BAD_COMMAND_ERROR_RETURN
0CB5 E8 0CD9 R                  call    CHECK_OP_NOT_A_RESET
0CB8 73 06                      jnb     short NOT_A_RESET
0CBA E8 0CE7 R                  call    RESET_CONTROLLER
0CBD EB 19                      jmp     short JUST_RETURN
0CBF 90                         nop
0CC0                    NOT_A_RESET:
0CC0 80 FC 09                   cmp     ah, INT13H_OP_09_INITIALIZE_DISK_TABLE
0CC3 75 06                      jne     short RUN_COMMAND
0CC5 E8 0D83 R                  call    INITIALIZE_DISK_TABLE
0CC8 EB 0E                      jmp     short JUST_RETURN
0CCA 90                         nop
0CCB                    RUN_COMMAND:
0CCB E8 0E0C R                  call    DO_COMMAND
0CCE EB 08                      jmp     short JUST_RETURN
0CD0 90                         nop
0CD1                    CONTROLLER_FUNC         ENDP

0CD1                    BAD_COMMAND_ERROR_RETURN        PROC NEAR
0CD1 C6 06 0474 R 01            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_01_BAD_COMMAND
0CD6 F9                         stc
0CD7 C3                         ret
0CD8                    BAD_COMMAND_ERROR_RETURN        ENDP

0CD8                    JUST_RETURN             PROC NEAR
0CD8 C3                         ret
0CD9                    JUST_RETURN             ENDP


                        ;--------------------------------------------------------
                        ;       Sets carry if the operation is a reset          :
                        ;--------------------------------------------------------

0CD9                    CHECK_OP_NOT_A_RESET    PROC NEAR
0CD9 80 FC 00                   cmp     ah, INT13H_OP_00_RESET_DISK_SYSTEM
0CDC 74 07                      je      short IS_A_RESET
0CDE 80 FC 0D                   cmp     ah, INT13H_OP_0D_ALTERNATE_DISK_RESET
0CE1 74 02                      je      short IS_A_RESET
0CE3 F8                         clc
0CE4 C3                         ret
0CE5                    IS_A_RESET:
0CE5 F9                         stc
0CE6 C3                         ret
0CE7                    CHECK_OP_NOT_A_RESET    ENDP


                        ;--------------------------------------------------------
                        ;       This issues a reset signal to the controller    :
                        ;       and then configures the drive geometry.         :
                        ;       Done in response to INT13H functions 00H        :
                        ;       ("Disk Subsystem Reset") and 0D ("Alternate     :
                        ;       Disk Reset"), but also on errors in hope of     :
                        ;       bringing the controller back to workable state. :
                        ;--------------------------------------------------------

0CE7                    RESET_CONTROLLER        PROC NEAR
0CE7 E8 0D61 R                  call    MASTER_RESET
0CEA E8 0D83 R                  call    INITIALIZE_DISK_TABLE
0CED C3                         ret
0CEE                    RESET_CONTROLLER        ENDP


                        ;--------------------------------------------------------
                        ;       Not sure why this exists or what command        :
                        ;       1AH would do.                                   :
                        ;--------------------------------------------------------

0CEE                    UNUSED_1A               PROC NEAR
0CEE C6 06 0442 R 1A            mov     BDA_CONTROLLER_DATA_BUFFER_00, 1Ah
0CF3 C7 06 0443 R 0000          mov     word ptr BDA_CONTROLLER_DATA_BUFFER_01, 0000H
0CF9 C6 06 0445 R 06            mov     BDA_CONTROLLER_DATA_BUFFER_03, 06H
0CFE C6 06 0446 R 03            mov     BDA_CONTROLLER_DATA_BUFFER_04, 03H
0D03 C6 06 0447 R 00            mov     BDA_CONTROLLER_DATA_BUFFER_05, 00H
0D08 E8 0E0C R                  call    DO_COMMAND
0D0B C3                         ret
0D0C                    UNUSED_1A               ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       This is the INT13H 01H ("Get Drive              ;
                        ;       Parameters") handler. What it just chews and    ;
                        ;       returns data that can be gotten straight from   ;
                        ;       the BDA anyways.                                ;
                        ;--------------------------------------------------------

0D0C                    INT13H_08_GET_DRIVE_PARAMS      PROC FAR
0D0C 5E                         pop     si
0D0D 57                         push    di
0D0E 1E                         push    ds
0D0F 06                         push    es
0D10 E8 1139 R                  call    ZERO_DS
                                ASSUME  ds:ZEROSEG
0D13 E8 0DFC R                  call    CHECK_VALID_DISK_NUMBER
0D16 73 0A                      jnb     short IS_A_VALID_DISK_NUMBER
0D18 C6 06 0474 R 01            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_01_BAD_COMMAND
0D1D B4 01                      mov     ah, 1
0D1F EB 2E                      jmp     short DONE_GETTING_PARAMS
0D21 90                         nop
0D22                    IS_A_VALID_DISK_NUMBER:
0D22 E8 0BFB R                  call    SET_BDA_STATUS_SUCCESS
0D25 B1 03                      mov     cl, 3
0D27 D2 CA                      ror     dl, cl
0D29 80 E2 60                   and     dl, 60h
0D2C 88 16 0443 R               mov     BDA_CONTROLLER_DATA_BUFFER_01, dl
0D30 E8 101E R                  call    DRIVE_PARAM_TO_DI
0D33 8A 16 0475 R               mov     dl, BDA_NUMBER_OF_HARD_DISKS
0D37 26: 8A 75 02               mov     dh, es:[di+2]
0D3B FE CE                      dec     dh
0D3D 26: 8B 05                  mov     ax, es:[di]
0D40 48                         dec     ax
0D41 48                         dec     ax
0D42 8A E8                      mov     ch, al
0D44 B1 06                      mov     cl, 6
0D46 D2 E4                      shl     ah, cl
0D48 86 E1                      xchg    ah, cl
0D4A 80 C9 11                   or      cl, 11h
0D4D 33 C0                      xor     ax, ax
0D4F                    DONE_GETTING_PARAMS:
0D4F 07                         pop     es
0D50 1F                         pop     ds
0D51 5F                         pop     di
0D52 CA 0002                    ret     2
0D55                    INT13H_08_GET_DRIVE_PARAMS      ENDP ; sp =     2


                        ;--------------------------------------------------------
                        ;       Not sure why this exists.                       :
                        ;--------------------------------------------------------

0D55                    UNUSED_BYTE_01          PROC NEAR
0D55 B1 03                      mov     cl, 3
0D57 D2 CA                      ror     dl, cl
0D59 80 E2 60                   and     dl, 60h
0D5C 88 16 0443 R               mov     BDA_CONTROLLER_DATA_BUFFER_01, dl
0D60 C3                         ret
0D61                    UNUSED_BYTE_01          ENDP


                        ;--------------------------------------------------------
                        ;       This issues a reset signal to the controller.   :
                        ;--------------------------------------------------------

0D61                    MASTER_RESET            PROC NEAR
0D61 BA 0321                    mov     dx, IO_PORT_321_READ_STATUS_WRITE_RESET
0D64 E8 103A R                  call    DO_NOTHING
0D67 E8 1137 R                  call    OUTB_DX_AL
0D6A B9 0684                    mov     cx, 684h                ; Arbitrary delay
0D6D                    DO_LOOP:
0D6D E2 FE                      loop    DO_LOOP
0D6F C3                         ret
0D70                    MASTER_RESET            ENDP


                        ;--------------------------------------------------------
                        ;       This is the INT13H 01H ("Get Disk Status")      :
                        ;       handler. It returns status of the last          :
                        ;       operation in AL and clears it.                  :
                        ;--------------------------------------------------------

0D70                    INT13H_01_GET_DISK_STATUS       PROC FAR
0D70 5E                         pop     si
0D71 32 E4                      xor     ah, ah
0D73 1E                         push    ds
0D74 E8 1139 R                  call    ZERO_DS
                                ASSUME  ds:ZEROSEG
0D77 A0 0474 R                  mov     al, BDA_LAST_OP_STATUS
0D7A 88 26 0474 R               mov     BDA_LAST_OP_STATUS, ah
0D7E 1F                         pop     ds
0D7F F8                         clc
0D80 CA 0002                    ret     2
0D83                    INT13H_01_GET_DISK_STATUS       ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Load disk geometry to the controller.           :
                        ;                                                       :
                        ;       This is done in response to INT13H op 09H       :
                        ;       ("Initialize Disk Table"), but also on reset    :
                        ;       and when dealing with error conditions.         :
                        ;                                                       :
                        ;       Note that this forces the head count to 2       :
                        ;       regardless of what's in the INT14H table;       :
                        ;       which happens to have head count of 4. Not      :
                        ;       sure why and the controller seems to happily    :
                        ;       accepts heads over 2 anyways.                   :
                        ;--------------------------------------------------------

0D83                    INITIALIZE_DISK_TABLE   PROC NEAR
0D83 C6 06 0442 R 0C                    mov     BDA_CONTROLLER_DATA_BUFFER_00, OP_0C_INIT_DRV_PARM
0D88 80 26 0443 R 00                    and     BDA_CONTROLLER_DATA_BUFFER_01, 00h      ; Start with the
                                                                                        ; first drive

0D8D                    START_INIT_DRV_PARM_COMMAND:
0D8D E8 0F39 R                          call    SELECT_BUSY_FOR_COMMAN
0D90 E8 0EFC R                          call    WRITE_COMMAND_BUFFER
0D93 73 09                              jnb     short COMMAND_ISSUED_SUCCESS

0D95                    BAD_TABLE_ERROR_RETURN:
0D95 E8 0D61 R                          call    MASTER_RESET
0D98 C6 06 0474 R 07                    mov     BDA_LAST_OP_STATUS, INT13H_STATUS_07_BAD_DISK_PARAM_TABLE
0D9D C3                                 ret

0D9E                    COMMAND_ISSUED_SUCCESS:
0D9E E8 0F45 R                          call    WAIT_FOR_BYTE_READY
0DA1 73 02                              jnb     short PROCEED_WRITING_DRIVE_PARAMS
0DA3 EB F0                              jmp     short BAD_TABLE_ERROR_RETURN

0DA5                    PROCEED_WRITING_DRIVE_PARAMS:
0DA5 06                                 push    es
0DA6 E8 1139 R                          call    ZERO_DS
                                        ASSUME  ds:ZEROSEG
0DA9 E8 101E R                          call    DRIVE_PARAM_TO_DI
0DAC 8B DF                              mov     bx, di
0DAE B9 0008                            mov     cx,                             ; Eight drive params bytes
0DB1 33 F6                              xor     si, si

0DB3                    WRITE_DRIVE_PARAM_BYTE:
0DB3 E8 0F45 R                          call    WAIT_FOR_BYTE_READY
0DB6 56                                 push    si
0DB7 53                                 push    bx
0DB8 BB 0DF4 R                          mov     bx, offset DATA_FROM_INT41H_OFFSETS
0DBB 2E: 8A 18                          mov     bl, cs:[bx+si]                  ; Read offset from the table
0DBE 32 FF                              xor     bh, bh
0DC0 8B F3                              mov     si, bx
0DC2 5B                                 pop     bx
0DC3 26: 8A 00                          mov     al, es:[bx+si]                  ; Read data from drive table
0DC6 83 FE 02                           cmp     si, 2                           ; Offset 2 is head number
0DC9 75 02                              jne     short OFFSET_NOT_TWO
0DCB B0 02                              mov     al, 2                           ; Force head number to two.
                                                                                ; WHY?
0DCD                    OFFSET_NOT_TWO:
0DCD BA 0320                            mov     dx, IO_PORT_320_DATA
0DD0 E8 103A R                          call    DO_NOTHING
0DD3 E8 1137 R                          call    OUTB_DX_AL
0DD6 5E                                 pop     si
0DD7 46                                 inc     si
0DD8 E0 D9                              loopne  WRITE_DRIVE_PARAM_BYTE
0DDA 07                                 pop     es
0DDB E8 0F6F R                          call    HANDLE_COMMAND_RESPONSE
0DDE 80 3E 0474 R 00                    cmp     BDA_LAST_OP_STATUS, INT13H_STATUS_00_NO_ERROR
0DE3 75 0E                              jne     short DONE_INITIALIZING_DISK_TABLE

0DE5 F6 06 0443 R 20                    test    BDA_CONTROLLER_DATA_BUFFER_01, 20h      ; Were we configuring
0DEA 75 07                              jne     short DONE_INITIALIZING_DISK_TABLE      ;  the second drive?

0DEC C6 06 0443 R 20                    mov     BDA_CONTROLLER_DATA_BUFFER_01, 20h      ; Proceed with the
0DF1 EB 9A                              jmp     short START_INIT_DRV_PARM_COMMAND       ;  second drive.

0DF3                    DONE_INITIALIZING_DISK_TABLE:
0DF3 C3                                 ret
0DF4                    INITIALIZE_DISK_TABLE   ENDP


                        ;--------------------------------------------------------
                        ;       Mapping from INT41H format to what the          :
                        ;       controller command 0CH ("Init drive             :
                        ;       parameters") expects. It essientially just      :
                        ;       flips endianness of WORD values.                :
                        ;--------------------------------------------------------

0DF4                    DATA_FROM_INT41H_OFFSETS:
0DF4 01                                 DB      1               ; Max cyls Hi
0DF5 00                                 DB      0               ; Max cyls Lo
0DF6 02                                 DB      2               ; Max heads
0DF7 04                                 DB      4               ; RPC Hi
0DF8 03                                 DB      3               ; RPC Lo
0DF9 06                                 DB      6               ; WPC Hi
0DFA 05                                 DB      5               ; WPC Lo
0DFB 07                                 DB      7               ; ECC Len
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Sets carry if not a drive 80H or 81H            :
                        ;--------------------------------------------------------

0DFC                    CHECK_VALID_DISK_NUMBER PROC NEAR
0DFC 80 EA 80                   sub     dl, 80h
0DFF 80 FA 02                   cmp     dl, 2
0E02 F5                         cmc
0E03 C3                         ret
0E04                    CHECK_VALID_DISK_NUMBER ENDP


                        ;--------------------------------------------------------
                        ;       Not sure why this exists...                     :
                        ;--------------------------------------------------------

0E04                    UNUSED_1B               PROC NEAR
0E04 C6 06 0442 R 1B            mov     BDA_CONTROLLER_DATA_BUFFER_00, 1Bh
0E09 EB 01                      jmp     short DO_COMMAND
0E0B 90                         nop
0E0C                    UNUSED_1B               ENDP


                        ;--------------------------------------------------------
                        ;       This is the command submission routine          :
                        ;       It sets up DMA if necessary, submits the        :
                        ;       command, reads the response and deals with      :
                        ;       the error sense if necessary.                   :
                        ;--------------------------------------------------------

0E0C                    DO_COMMAND              PROC NEAR
0E0C E8 0E9E R                  call    CHECK_COMMAND_USES_DMA
0E0F 73 68                      jnc     short ISSUE_THE_COMMAND ; Just issue it, no DMA
0E11 06                         push    es
0E12 58                         pop     ax
0E13 B1 04                      mov     cl, 4
0E15 D3 E0                      shl     ax, cl
0E17 03 C3                      add     ax, bx
0E19 B9 FFFF                    mov     cx, 0FFFFh              ; Maximum ISA DMA length -- 64K
0E1C 2B C8                      sub     cx, ax
0E1E 91                         xchg    ax, cx
0E1F E8 0ED2 R                  call    SET_DATA_LENGTH
0E22 3B C1                      cmp     ax, cx
0E24 73 06                      jnb     short DMA_ADDRESS_FINE
0E26                    BAD_DMA:
0E26 C6 06 0474 R 09            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_09_DMA_ACROSS_64K
0E2B C3                         ret

0E2C                    DMA_ADDRESS_FINE:
0E2C B0 80                      mov     al, 128
0E2E E8 0EEA R                  call    NOT_A_LONG_COMMAND
0E31 73 02                      jnb     short CONFIGURE_DMA
0E33 B0 7F                      mov     al, 127
0E35                    CONFIGURE_DMA:
0E35 38 06 0446 R               cmp     BDA_CONTROLLER_DATA_BUFFER_04, al
0E39 77 EB                      ja      short BAD_DMA

0E3B FA                         cli
0E3C E8 0EBD R                  call    SELECT_DMA_MODE         ; IN or OUT
0E3F E6 0B                      out     0Bh, al                 ; Set DMA mode
0E41 EB 00                      jmp     short $+2
0E43 E6 0C                      out     0Ch, al                 ; Clear DMA counter

0E45 06                         push    es
0E46 58                         pop     ax
0E47 8B C8                      mov     cx, ax
0E49 B1 04                      mov     cl, 4
0E4B D2 ED                      shr     ch, cl
0E4D D3 E0                      shl     ax, cl
0E4F 03 C3                      add     ax, bx
0E51 80 D5 00                   adc     ch, 0
0E54 BA 0006                    mov     dx, 6
0E57 EE                         out     dx, al                  ; DMA base bits 0-7
0E58 86 C4                      xchg    al, ah
0E5A EE                         out     dx, al                  ; DMA base bits 8-15
0E5B 86 C5                      xchg    al, ch
0E5D BA 0082                    mov     dx, 82h                 ; Page register
0E60 EE                         out     dx, al                  ; DMA base bits 16-23

0E61 E8 0ED2 R                  call    SET_DATA_LENGTH
0E64 8B C1                      mov     ax, cx
0E66 BA 0007                    mov     dx, 7
0E69 EE                         out     dx, al                  ; Word count low
0E6A 86 E0                      xchg    ah, al
0E6C EE                         out     dx, al                  ; Word count high

0E6D FB                         sti
0E6E B0 01                      mov     al, 1                   ; IRQEN=0 DMAEN=1 -- Enable DMA
0E70 BA 0323                    mov     dx, IO_PORT_323_DMA_IRQ
0E73 E8 103A R                  call    DO_NOTHING
0E76 E8 1137 R                  call    OUTB_DX_AL

0E79                    ISSUE_THE_COMMAND:
0E79 E8 0F39 R                  call    SELECT_BUSY_FOR_COMMAND
0E7C E8 0EFC R                  call    WRITE_COMMAND_BUFFER
0E7F 9C                         pushf
0E80 80 26 0447 R 3F            and     BDA_CONTROLLER_DATA_BUFFER_05, 3Fh
0E85 9D                         popf
0E86 73 09                      jnb     short COMMAND_REQUIRES_RESPONSE
0E88 E8 0D83 R                  call    INITIALIZE_DISK_TABLE
0E8B C6 06 0474 R 80            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_80_TIMEOUT
0E90 C3                         ret
0E91                    COMMAND_REQUIRES_RESPONSE:
0E91 E8 0E9E R                  call    CHECK_COMMAND_USES_DMA
0E94 73 04                      jnb     short READ_RESPONSE
0E96 B0 03                      mov     al, 011b                ; Unmask DRQ3
0E98 E6 0A                      out     0Ah, al
0E9A                    READ_RESPONSE:
0E9A E8 0F6F R                  call    HANDLE_COMMAND_RESPONSE
0E9D C3                         ret
0E9E                    DO_COMMAND              ENDP


```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Sets carry if the command requires a DMA        :
                        ;       transfer for the data                           :
                        ;--------------------------------------------------------

0E9E                    CHECK_COMMAND_USES_DMA  PROC NEAR
0E9E 56                         push    si
0E9F 51                         push    cx
0EA0 50                         push    ax
0EA1 BE 1224 R                  mov     si, offset DMA_COMMAND_TABLE
0EA4 B9 0006                    mov     cx, 6
0EA7 90                         nop
0EA8 A0 0442 R                  mov     al, BDA_CONTROLLER_DATA_BUFFER_00
0EAB                    CHECK_DMA_COMMAND_TABLE_ENTRY:
0EAB 2E: 3A 04                  cmp     al, cs:[si]
0EAE 74 08                      je      short COMMAND_USES_DMA
0EB0 46                         inc     si
0EB1 E2 F8                      loop    CHECK_DMA_COMMAND_TABLE_ENTRY
0EB3 58                         pop     ax
0EB4 59                         pop     cx
0EB5 5E                         pop     si
0EB6 F8                         clc
0EB7 C3                         ret
0EB8                    COMMAND_USES_DMA:
0EB8 58                         pop     ax
0EB9 59                         pop     cx
0EBA 5E                         pop     si
0EBB F9                         stc
0EBC C3                         ret
0EBD                    CHECK_COMMAND_USES_DMA  ENDP


                        ;--------------------------------------------------------
                        ;       Sets AL to DMA mode byte appropriate for        :
                        ;       given command, depending on the direction       :
                        ;       of the transfer necessary.                      :
                        ;--------------------------------------------------------

0EBD                    SELECT_DMA_MODE         PROC NEAR
0EBD A0 0442 R                  mov     al, BDA_CONTROLLER_DATA_BUFFER_00
0EC0 3C 08                      cmp     al, OP_08_READ_SECTORS
0EC2 74 0B                      je      short READ_COMMAND
0EC4 3C 0E                      cmp     al, OP_0E_READ_SECTOR_BUFFER
0EC6 74 07                      je      short READ_COMMAND
0EC8 3C E5                      cmp     al, OP_E5_READ_LONG
0ECA 74 03                      je      short READ_COMMAND
0ECC                    WRITE_COMMAND:
0ECC B0 4B                      mov     al, DRQ3_WRITE
0ECE C3                         ret
0ECF                    READ_COMMAND:
0ECF B0 47                      mov     al, DRQ3_READ
0ED1 C3                         ret
0ED2                    SELECT_DMA_MODE         ENDP


                        ;--------------------------------------------------------
                        ;       Sets CX to 518 if the command is                :
                        ;       Read/Write Long, 512 otherwise.                 :
                        ;--------------------------------------------------------

0ED2                    SET_DATA_LENGTH         PROC NEAR
0ED2 50                         push    ax
0ED3 B9 0206                    mov     cx, 518                 ; Long command length: 512 + 5 (ecc?)
0ED6 E8 0EEA R                  call    NOT_A_LONG_COMMAND
0ED9 72 03                      jc      short DATA_LENGTH_SET
0EDB B9 0200                    mov     cx, 512                 ; Regular command length
0EDE                    DATA_LENGTH_SET:
0EDE 32 E4                      xor     ah, ah
0EE0 A0 0446 R                  mov     al, BDA_CONTROLLER_DATA_BUFFER_04
0EE3 F7 E1                      mul     cx
0EE5 48                         dec     ax
0EE6 8B C8                      mov     cx, ax
0EE8 58                         pop     ax
0EE9 C3                         ret
0EEA                    SET_DATA_LENGTH         ENDP


                        ;--------------------------------------------------------
                        ;       Sets carry if the command is                    :
                        ;       Read Long or Write Long                         :
                        ;--------------------------------------------------------

0EEA                    NOT_A_LONG_COMMAND      PROC NEAR
0EEA 80 3E 0442 R E5            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_E5_READ_LONG
0EEF 75 02                      jne     short NOT_READ_LONG
0EF1                    READ_LONG_OR_WRITE_LONG:
0EF1 F9                         stc
0EF2 C3                         ret
0EF3                    NOT_READ_LONG:
0EF3 80 3E 0442 R E6            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_E6_WRITE_LONG
0EF8 74 F7                      je      short READ_LONG_OR_WRITE_LONG
0EFA F8                         clc
0EFB C3                         ret
0EFC                    NOT_A_LONG_COMMAND      ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Write the while 6-byte command buffer           :
                        ;       Sets carry on failure (timeout)                 :
                        ;--------------------------------------------------------

0EFC                    WRITE_COMMAND_BUFFER    PROC NEAR
0EFC E8 0F19 R                  call    WAIT_READY_FOR_COMMAND
0EFF 73 01                      jnb     short READY_WAIT_SUCCESSFUL
0F01 C3                         ret
0F02                    READY_WAIT_SUCCESSFUL:
0F02 BE 0442 R                  mov     si, offset BDA_CONTROLLER_DATA_BUFFER_00
0F05 33 C9                      xor     cx, cx
0F07 BA 0320                    mov     dx, IO_PORT_320_DATA
0F0A E8 103A R                  call    DO_NOTHING
0F0D                    WRITE_COMMAND_BYTE:
0F0D 8A 04                      mov     al, [si]
0F0F EE                         out     dx, al
0F10 46                         inc     si
0F11 FE C1                      inc     cl
0F13 80 F9 06                   cmp     cl, 6
0F16 75 F5                      jne     short WRITE_COMMAND_BYTE
0F18 C3                         ret
0F19                    WRITE_COMMAND_BUFFER    ENDP


                        ;--------------------------------------------------------
                        ;       This waits until the controller is ready        :
                        ;       to accept a command byte.                       :
                        ;                                                       :
                        ;       Sets carry on failure (timeout)                 :
                        ;--------------------------------------------------------

0F19                    WAIT_READY_FOR_COMMAND  PROC NEAR
0F19 B9 FFFF                    mov     cx, 0FFFFh              ; Try 0xffff times
0F1C BA 0321                    mov     dx, IO_PORT_321_READ_STATUS_WRITE_RESET
0F1F E8 103A R                  call    DO_NOTHING
0F22                    SEE_IF_READY:
0F22 E8 1135 R                  call    INB_AL_DX
0F25 24 0D                      and     al, 1101b               ; BUSY=1 COMMAND0/DATA1=1
                                                                ;  INPUT1/OUTPUT0=0 REQ_READY=1
0F27 3C 0D                      cmp     al, 1101b               ; The controller is ready to
                                                                ;  accept a command byte.
0F29 75 02                      jne     short NOT_READY_YET
0F2B F8                         clc
0F2C C3                         ret
0F2D                    NOT_READY_YET:
0F2D E2 F3                      loop    SEE_IF_READY
0F2F C6 06 0474 R 80            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_80_TIMEOUT
0F34 E8 0D61 R                  call    MASTER_RESET
0F37 F9                         stc
0F38 C3                         ret
0F39                    WAIT_READY_FOR_COMMAND  ENDP


                        ;--------------------------------------------------------
                        ;       Issue a select signal.                          :
                        ;       This has to be done to start a command.         :
                        ;       The controller responds by enabling BUSY        :
                        ;       in status byte.                                 :
                        ;--------------------------------------------------------

0F39                    SELECT_BUSY_FOR_COMMAND PROC NEAR
0F39 BA 0322                    mov     dx, IO_PORT_322_READ_CONFIG_WRITE_SELECT
0F3C E8 103A R                  call    DO_NOTHING
0F3F B0 01                      mov     al, 1                   ; Value is ignored
0F41 E8 1137 R                  call    OUTB_DX_AL
0F44 C3                         ret
0F45                    SELECT_BUSY_FOR_COMMAND ENDP


                        ;--------------------------------------------------------
                        ;       Wait until a byte can be read                   :
                        ;       from the contoller                              :
                        ;                                                       :
                        ;       Sets carry on failure (timeout)                 :
                        ;--------------------------------------------------------

0F45                    WAIT_FOR_BYTE_READY     PROC NEAR
0F45 51                         push    cx
0F46 52                         push    dx
0F47 50                         push    ax
0F48 B9 001E                    mov     cx, 30                  ; 30 x 0xffff times...

0F4B                    OUTER_CHECK_READY:
0F4B 51                         push    cx
0F4C B9 FFFF                    mov     cx, 0FFFFh              ; ...30 x 0xffff times
0F4F BA 0321                    mov     dx, IO_PORT_321_READ_STATUS_WRITE_RESET
0F52 E8 103A R                  call    DO_NOTHING
0F55                    INNER_CHECK_READY:
0F55 E8 1135 R                  call    INB_AL_DX
0F58 24 01                      and     al, 1                   ; REQUEST=1
0F5A 75 0E                      jnz     short SUCCESSFULLY_READY
0F5C E2 F7                      loop    INNER_CHECK_READY
0F5E 59                         pop     cx
0F5F E2 EA                      loop    OUTER_CHECK_READY

0F61 C6 06 0474 R 80            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_80_TIMEOUT
0F66 F9                         stc
0F67 EB 02                      jmp     short DONE_WAITING
0F69 90                         nop

0F6A                    SUCCESSFULLY_READY:
0F6A 59                         pop     cx
0F6B                    DONE_WAITING:
0F6B 58                         pop     ax
0F6C 5A                         pop     dx
0F6D 59                         pop     cx
0F6E C3                         ret
0F6F                    WAIT_FOR_BYTE_READY     ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Wait for command to finish with an              :
                        ;       appropriate timeout. Then check status byte     :
                        ;       and possibly handle an error. Disable DMA       :
                        ;       on disk and DMA controllers afterwards.         :
                        ;--------------------------------------------------------

0F6F                    HANDLE_COMMAND_RESPONSE PROC NEAR
0F6F 06                         push    es
0F70 E8 1139 R                  call    ZERO_DS
                                ASSUME  ds:ZEROSEG
0F73 E8 101E R                  call    DRIVE_PARAM_TO_DI
0F76 26: 8A 4D 0A               mov     cl, es:[di+(TIMEOUT_FMT - INT41H_DATA)]
0F7A 80 3E 0442 R 04            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_04_FORMAT_DRIVE
0F7F 74 0F                      je      short CL_TIMEOUT_SET
0F81 26: 8A 4D 0B               mov     cl, es:[di+(TIMEOUT_CHK - INT41H_DATA)]
0F85 80 3E 0442 R E3            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_E3_DRIVE_DIAG
0F8A 74 04                      je      short CL_TIMEOUT_SET
0F8C 26: 8A 4D 09               mov     cl, es:[di+(TIMEOUT_STD - INT41H_DATA)]
0F90                    CL_TIMEOUT_SET:
0F90 07                         pop     es

0F91 32 ED                      xor     ch, ch
0F93 B8 0444                    mov     ax, 444h
0F96 F7 E1                      mul     cx
0F98 52                         push    dx
0F99 E8 1028 R                  call    BDA_TIMER_TO_CX_DX
0F9C 03 D0                      add     dx, ax
0F9E 83 D1 00                   adc     cx, 0
0FA1 5B                         pop     bx
0FA2 03 CB                      add     cx, bx
0FA4 8B D9                      mov     bx, cx
0FA6 8B C2                      mov     ax, dx

0FA8                    CHECK_BYTE_READY_FOR_READ:
0FA8 50                         push    ax
0FA9 53                         push    bx
0FAA BA 0321                    mov     dx, IO_PORT_321_READ_STATUS_WRITE_RESET
0FAD E8 103A R                  call    DO_NOTHING
0FB0 E8 1135 R                  call    INB_AL_DX
0FB3 24 07                      and     al, 111b                        ; BUSY=0 CONTROL0/DATA1=1
                                                                        ;  INPUT1/OUTPUT0=1 REQ_BYTE=1
0FB5 3C 07                      cmp     al, 111b                        ;  Is data byte ready for
                                                                        ;  input to host?
0FB7 75 05                      jne     short DATA_BYTE_NOT_READY
0FB9 5B                         pop     bx
0FBA 58                         pop     ax
0FBB EB 13                      jmp     short WAIT_DONE
0FBD 90                         nop
0FBE                    DATA_BYTE_NOT_READY:
0FBE E8 1028 R                  call    BDA_TIMER_TO_CX_DX
0FC1 5B                         pop     bx
0FC2 58                         pop     ax
0FC3 3B D9                      cmp     bx, cx
0FC5 77 E1                      ja      short CHECK_BYTE_READY_FOR_READ
0FC7 72 2C                      jb      short TIMED_OUT
0FC9 3B C2                      cmp     ax, dx
0FCB 77 DB                      ja      short CHECK_BYTE_READY_FOR_READ
0FCD EB 26                      jmp     short TIMED_OUT
0FCF 90                         nop
0FD0                    WAIT_DONE:
0FD0 B0 07                      mov     al, 111b                        ; Mask DRQ3
0FD2 E6 0A                      out     0Ah, al

0FD4 32 C0                      xor     al, al                          ; Disable DMA and IRQ
0FD6 BA 0323                    mov     dx, IO_PORT_323_DMA_IRQ
0FD9 E8 103A R                  call    DO_NOTHING
0FDC E8 1137 R                  call    OUTB_DX_AL

0FDF BA 0320                    mov     dx, IO_PORT_320_DATA
0FE2 E8 103A R                  call    DO_NOTHING
0FE5 E8 1135 R                  call    INB_AL_DX                       ; Read the status byte
0FE8 24 02                      and     al, 2
0FEA A2 0474 R                  mov     BDA_LAST_OP_STATUS, al
0FED 3C 00                      cmp     al, 0
0FEF 74 03                      je      short DONE_HANDLING_RESPONSE    ; No error
0FF1 E8 103B R                  call    READ_ERROR_SENSE                ; Error encountered, read sense
0FF4                    DONE_HANDLING_RESPONSE:
0FF4 C3                         ret

0FF5                    TIMED_OUT:
0FF5 B0 07                      mov     al, 111b                        ; Mask DRQ3
0FF7 E6 0A                      out     0Ah, al

0FF9 32 C0                      xor     al, al                          ; Disable DMA and IRQ
0FFB BA 0323                    mov     dx, IO_PORT_323_DMA_IRQ
0FFE E8 103A R                  call    DO_NOTHING
1001 E8 1137 R                  call    OUTB_DX_AL

1004 80 3E 0442 R 0C            cmp     BDA_CONTROLLER_DATA_BUFFER_00, OP_0C_INIT_DRV_PARM
1009 74 0A                      je      short RETURN_ERROR_BAD_DISK_TABLE
100B E8 0CE7 R                  call    RESET_CONTROLLER
100E C6 06 0474 R 80            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_80_TIMEOUT
1013 EB 08                      jmp     short FINISHED_HANDLING_RESPONSE

1015                    RETURN_ERROR_BAD_DISK_TABLE:
1015 E8 0D61 R                  call    MASTER_RESET
1018 C6 06 0474 R 07            mov     BDA_LAST_OP_STATUS, INT13H_STATUS_07_BAD_DISK_PARAM_TABLE

101D                    FINISHED_HANDLING_RESPONSE:
101D C3                         ret
101E                    HANDLE_COMMAND_RESPONSE ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Just gets the pointer to INT41H.                :
                        ;--------------------------------------------------------

101E                    DRIVE_PARAM_TO_DI       PROC NEAR
101E 1E                                 push    ds
101F E8 1141 R                          call    ZERO_DS_ALT
                                        ASSUME  ds:ZEROSEG
1022 C4 3E 0104 R                       les     di, dword ptr INT41H_OFFSET
1026 1F                                 pop     ds
1027 C3                                 ret
1028                    DRIVE_PARAM_TO_DI       ENDP

                        ;--------------------------------------------------------
                        ;       Read the timer. Used for delays.                :
                        ;--------------------------------------------------------

1028                    BDA_TIMER_TO_CX_DX      PROC NEAR
1028 50                                 push    ax
1029 1E                                 push    ds
102A E8 1139 R                          call    ZERO_DS
                                        ASSUME  ds:ZEROSEG
102D FA                                 cli
102E 8B 16 046C R                       mov     dx, BDA_TIMER_COUNTER_LO
1032 8B 0E 046E R                       mov     cx, BDA_TIMER_COUNTER_HI
1036 FB                                 sti
1037 1F                                 pop     ds
1038 58                                 pop     ax
1039 C3                                 ret
103A                    BDA_TIMER_TO_CX_DX      ENDP

                        ;--------------------------------------------------------
                        ;       God knows why does this exist. Perhaps it       :
                        ;       is makes it easier to hook a tracing routine.   :
                        ;--------------------------------------------------------

103A                    DO_NOTHING              PROC NEAR
103A C3                                 ret
103B                    DO_NOTHING              ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       This routine is called when a command           :
                        ;       returns an error status.                        :
                        ;       It reads in the error details from controler    :
                        ;       and tries to figure out the proper INT13H       :
                        ;       error code.                                     :
                        ;--------------------------------------------------------

103B                    READ_ERROR_SENSE        PROC NEAR
103B E8 1139 R                          call    ZERO_DS
                                        ASSUME  ds:ZEROSEG
103E C6 06 0442 R 03                    mov     BDA_CONTROLLER_DATA_BUFFER_00, OP_03_READ_SENSE
1043 E8 0F39 R                          call    SELECT_BUSY_FOR_COMMAND
1046 E8 0EFC R                          call    WRITE_COMMAND_BUFFER
1049 72 36                              jc      short SENSE_FAILED
104B E8 0F45 R                          call    WAIT_FOR_BYTE_READY
104E 72 31                              jc      short SENSE_FAILED

1050 33 C9                              xor     cx, cx
1052 BE 0000                            mov     si, 0
1055 BA 0320                            mov     dx, IO_PORT_320_DATA
1058 E8 103A R                          call    DO_NOTHING
105B                    READ_SENSE_BYTE:
105B E8 0F45 R                          call    WAIT_FOR_BYTE_READY
105E 72 21                              jc      short SENSE_FAILED
1060 E8 1135 R                          call    INB_AL_DX
1063 88 84 0442 R                       mov     BDA_CONTROLLER_DATA_BUFFER_00[si], al
1067 46                                 inc     si
1068 FE C1                              inc     cl
106A 80 F9 04                           cmp     cl, 4
106D 75 EC                              jne     short READ_SENSE_BYTE

106F E8 0F45 R                          call    WAIT_FOR_BYTE_READY
1072 72 0D                              jc      short SENSE_FAILED
1074 BA 0320                            mov     dx, IO_PORT_320_DATA
1077 E8 103A R                          call    DO_NOTHING
107A E8 1135 R                          call    INB_AL_DX
107D 24 02                              and     al, 2
107F 74 09                              je      short XLATE_SENSE_TO_INT13H_STATUS
1081                    SENSE_FAILED:
1081 E8 0CE7 R                          call    RESET_CONTROLLER
1084 C6 06 0474 R FF                    mov     BDA_LAST_OP_STATUS, INT13H_STATUS_FF_SENSE_OP_FAILED
1089 C3                                 ret

108A                    XLATE_SENSE_TO_INT13H_STATUS:
108A 8A 1E 0442 R                       mov     bl, BDA_CONTROLLER_DATA_BUFFER_00
108E 8A FB                              mov     bh, bl
1090 B1 04                              mov     cl, 4
1092 D2 EB                              shr     bl, cl
1094 80 E3 03                           and     bl, 3
1097 BE 1111 R                          mov     si, offset SENSE_ERR_0x
109A 80 FB 00                           cmp     bl, 0
109D 74 16                              je      short DO_XLATE_SENSE
109F 81 C6 000A                         add     si, SENSE_ERR_1x - SENSE_ERR_0x
10A3 80 FB 01                           cmp     bl, 1
10A6 74 0D                              je      short DO_XLATE_SENSE
10A8 81 C6 0010                         add     si, SENSE_ERR_2x - SENSE_ERR_1x
10AC 80 FB 02                           cmp     bl, 2
10AF 74 04                              je      short DO_XLATE_SENSE
10B1 81 C6 0004                         add     si, SENSE_ERR_3x - SENSE_ERR_2x
10B5                    DO_XLATE_SENSE:
10B5 80 E7 0F                           and     bh, 0Fh
10B8 53                                 push    bx
10B9 8A C7                              mov     al, bh
10BB 32 FF                              xor     bh, bh
10BD 2E: 3A 87 1131 R                   cmp     al, cs:SENSE_ERR_HI[bx]
10C2 5B                                 pop     bx
10C3 72 06                              jb      short XLATE_SUCCESSFUL
10C5 C6 06 0474 R BB                    mov     BDA_LAST_OP_STATUS, INT13H_STATUS_BB_UNDEFINED_ERROR
10CA C3                                 ret
10CB                    XLATE_SUCCESSFUL:
10CB 86 FB                              xchg    bh, bl
10CD 32 FF                              xor     bh, bh
10CF 2E: 8A 00                          mov     al, cs:[bx+si]
10D2 A2 0474 R                          mov     BDA_LAST_OP_STATUS, al
10D5 8A 0E 0442 R                       mov     cl, BDA_CONTROLLER_DATA_BUFFER_00
10D9 80 E1 1F                           and     cl, 1Fh
10DC 80 F9 18                           cmp     cl, 18h                         ; 018H = Correctable ECC error
10DF 75 2F                              jne     short DONE_READING_SENSE
10E1 C6 06 0442 R 0D                    mov     BDA_CONTROLLER_DATA_BUFFER_00, OP_0D_READ_ECC_BURST_ERROR_LEN
10E6 E8 0F39 R                          call    SELECT_BUSY_FOR_COMMAND
10E9 E8 0EFC R                          call    WRITE_COMMAND_BUFFER
10EC 73 07                              jnb     short READ_ECC_COMMAND_OKAY
10EE                    ERROR_READING_SENSE:
10EE E8 0CE7 R                          call    RESET_CONTROLLER
10F1 F9                                 stc
10F2 EB 1C                              jmp     short DONE_READING_SENSE
10F4 90                                 nop

10F5                    READ_ECC_COMMAND_OKAY:
10F5 E8 0F45 R                          call    WAIT_FOR_BYTE_READY
10F8 72 F4                              jc      short ERROR_READING_SENSE
10FA BA 0320                            mov     dx, IO_PORT_320_DATA
10FD E8 103A R                          call    DO_NOTHING
1100 E8 1135 R                          call    INB_AL_DX
1103 50                                 push    ax
1104 E8 0F45 R                          call    WAIT_FOR_BYTE_READY
1107 73 03                              jnb     short READ_ECC_READY
1109 58                                 pop     ax
110A EB E2                              jmp     short ERROR_READING_SENSE

110C                    READ_ECC_READY:
110C E8 1135 R                          call    INB_AL_DX
110F 58                                 pop     ax
1110                    DONE_READING_SENSE:
1110 C3                                 ret
1111                    READ_ERROR_SENSE        ENDP

```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       Sparse table for mapping senser error byte      :
                        ;       from the controller to INT13H error code        :
                        ;--------------------------------------------------------

1111                    SENSE_ERR_0x:
1111 00                         DB      INT13H_STATUS_00_NO_ERROR               ; 00
1112 20                         DB      INT13H_STATUS_20_CTRLR_ERROR            ; 01
1113 40                         DB      INT13H_STATUS_40_SEEK_FAILURE           ; 02
1114 20                         DB      INT13H_STATUS_20_CTRLR_ERROR            ; 03
1115 80                         DB      INT13H_STATUS_80_TIMEOUT                ; 04
1116 BB                         DB      INT13H_STATUS_BB_UNDEFINED_ERROR        ; 05
1117 20                         DB      INT13H_STATUS_20_CTRLR_ERROR            ; 06
1118 BB                         DB      INT13H_STATUS_BB_UNDEFINED_ERROR        ; 07
1119 40                         DB      INT13H_STATUS_40_SEEK_FAILURE           ; 08
111A 27                         DB      INT13H_STATUS_27_NEED_RECALIBRATE       ; 09
111B                    SENSE_ERR_1x:
111B 10                         DB      INT13H_STATUS_10_ECC_ERROR              ; 10
111C 10                         DB      INT13H_STATUS_10_ECC_ERROR              ; 11
111D 02                         DB      INT13H_STATUS_02_ADDR_MARK_NOT_FOUND    ; 12
111E 02                         DB      INT13H_STATUS_02_ADDR_MARK_NOT_FOUND    ; 13
111F 04                         DB      INT13H_STATUS_04_SECTOR_NOT_FOUND       ; 14
1120 40                         DB      INT13H_STATUS_40_SEEK_FAILURE           ; 15
1121 20                         DB      INT13H_STATUS_20_CTRLR_ERROR            ; 16
1122 29                         DB      29h                                     ; 17
1123 11                         DB      INT13H_STATUS_11_ECC_FIXED              ; 18
1124 0B                         DB      INT13H_STATUS_0B_BAD_CYLINDER           ; 19
1125 21                         DB      21h                                     ; 1A
1126 BB                         DB      INT13H_STATUS_BB_UNDEFINED_ERROR        ; 1B
1127 22                         DB      22h                                     ; 1C
1128 23                         DB      23h                                     ; 1D
1129 24                         DB      24h                                     ; 1E
112A 25                         DB      25h                                     ; 1D
112B                    SENSE_ERR_2x:
112B 01                         DB      INT13H_STATUS_01_BAD_COMMAND            ; 20
112C 02                         DB      INT13H_STATUS_02_ADDR_MARK_NOT_FOUND    ; 21
112D 01                         DB      INT13H_STATUS_01_BAD_COMMAND            ; 22
112E 04                         DB      INT13H_STATUS_04_SECTOR_NOT_FOUND       ; 23
112F                    SENSE_ERR_3x:
112F 20                         DB      INT13H_STATUS_20_CTRLR_ERROR            ; 30
1130 20                         DB      INT13H_STATUS_20_CTRLR_ERROR            ; 31
1131                    SENSE_ERR_END:

1131 0A                 SENSE_ERR_HI    DB      SENSE_ERR_1x - SENSE_ERR_0x
1132 10                         DB      SENSE_ERR_2x - SENSE_ERR_1x
1133 04                         DB      SENSE_ERR_3x - SENSE_ERR_2x
1134 02                         DB      SENSE_ERR_END - SENSE_ERR_3x

                        ;--------------------------------------------------------
                        ;       I/O routines.                                   :
                        ;                                                       :
                        ;       Separate subroutines seem to make little        :
                        ;       sense given the call insn is larger than        :
                        ;       IN/OUT alone. Perhaps this is useful for        :
                        ;       hooking up trace routines or something...       :
                        ;--------------------------------------------------------

1135                    INB_AL_DX               PROC NEAR
1135 EC                         in      al, dx
1136 C3                         ret
1137                    INB_AL_DX               ENDP

1137                    OUTB_DX_AL              PROC NEAR
1137 EE                         out     dx, al
1138 C3                         ret
1139                    OUTB_DX_AL              ENDP


                        ;--------------------------------------------------------
                        ;       Two different ways to set DS to zero            :
                        ;       Not sure why...                                 :
                        ;--------------------------------------------------------

1139                    ZERO_DS                 PROC NEAR
1139 50                         push    ax
113A B8 ---- R                  mov     ax, ZEROSEG
113D 50                         push    ax
113E 1F                         pop     ds
113F 58                         pop     ax
1140 C3                         ret
1141                    ZERO_DS                 ENDP

1141                    ZERO_DS_ALT             PROC NEAR
1141 50                         push    ax
1142 33 C0                      xor     ax, ax
1144 8E D8                      mov     ds, ax
1146 58                         pop     ax
1147 C3                         ret
1148                    ZERO_DS_ALT             ENDP


```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       OS Boot Sector load service (INT 19H)           :
                        ;--------------------------------------------------------

1148                    INT19H_HANDLER          PROC NEAR
1148 BA 0000                            mov     dx, 0                   ; Try the floppy
114B E8 1175 R                          call    READ_BOOT_SECTOR
114E 73 20                              jnb     short DO_THE_BOOT
1150 BA 0080                            mov     dx, 80h                 ; Try the hard drive
1153 E8 1175 R                          call    READ_BOOT_SECTOR
1156 73 0F                              jnb     short CHECK_BOOT_SIGNATURE
1158                    BOOT_FAILED:
1158 BE 11AD R                          mov     si, offset STRING_HARD_DISK_IS_OFF
115B B9 0061                            mov     cx, 61h                 ; strlen boot failed
115E 90                                 nop
115F 0E                                 push    cs
1160 1F                                 pop     ds
                                        ASSUME  ds:ROM
1161 E8 0B3C R                          call    PRINT_CX_CHARS_FROM_DS_SI
1164                    INFINITE_LOOP:
1164 90                                 nop
1165 EB FD                              jmp     short INFINITE_LOOP
1167                    CHECK_BOOT_SIGNATURE:
1167 26: 81 3E 7DFE AA55                cmp     word ptr es:7DFEh, 0AA55h
116E 75 E8                              jne     short BOOT_FAILED

1170                    DO_THE_BOOT:
1170 EA 7C00 ---- R                     jmp     BOOT_SEC
1175                    INT19H_HANDLER          ENDP

                        ;--------------------------------------------------------
                        ;       Expects DX=disk number                          :
                        ;       Sets carry on error                             :
                        ;--------------------------------------------------------

1175                    READ_BOOT_SECTOR        PROC NEAR
1175 33 C0                              xor     ax, ax
1177 8E C0                              mov     es, ax
1179 BB 7C00 R                          mov     bx, offset BOOT_SEC     ; Target address
117C B9 0002                            mov     cx, 2                   ; Number of attempts
117F                    TRY_RESET:
117F 51                                 push    cx
1180 B8 0000                            mov     ax, 0                   ; Reset disk system
1183 CD 13                              int     13h
1185 59                                 pop     cx
1186 73 0F                              jnb     short RESET_SUCCESSFUL
1188 80 FC 80                           cmp     ah, 80h
118B 75 04                              jne     short RESET_ANOTHER
118D F9                                 stc
118E EB 1C                              jmp     short DONE_READING_BOOT_SECTOR
1190 90                                 nop
1191                    RESET_ANOTHER:
1191 E2 EC                              loop    TRY_RESET
1193 F9                                 stc
1194 EB 16                              jmp     short DONE_READING_BOOT_SECTOR
1196 90                                 nop
1197                    RESET_SUCCESSFUL:
1197 B9 0002                            mov     cx, 2                   ; Two attempts
119A                    TRY_READ_SECTOR:
119A 51                                 push    cx
119B B9 0001                            mov     cx, 0001h               ; Track 0, sector 1
119E B8 0201                            mov     ax, 0201h               ; 02h = read, 01h sectors
11A1 CD 13                              int     13h
11A3 59                                 pop     cx
11A4 80 FC 00                           cmp     ah, 0                   ; Check status
11A7 74 03                              je      short DONE_READING_BOOT_SECTOR
11A9 E2 EF                              loop    TRY_READ_SECTOR
11AB F9                                 stc
11AC                    DONE_READING_BOOT_SECTOR:
11AC C3                                 ret
11AD                    READ_BOOT_SECTOR        ENDP

11AD                    STRING_HARD_DISK_IS_OFF:
11AD 20 20 48 61 72 64                  DB      "  Hard disk is off.",0Dh,0Ah
     20 64 69 73 6B 20
     69 73 20 6F 66 66
     2E 0D 0A
11C2 20 20 53 65 74 20                  DB      "  Set HDD switch to on or",0Dh,0Ah
     48 44 44 20 73 77
     69 74 63 68 20 74
     6F 20 6F 6E 20 6F
     72 0D 0A
11DD 20 20 49 6E 73 65                  DB      "  Insert DOS disk in drive and RESET the System. "
     72 74 20 44 4F 53
     20 64 69 73 6B 20
     69 6E 20 64 72 69
     76 65 20 61 6E 64
     20 52 45 53 45 54
     20 74 68 65 20 53
     79 73 74 65 6D 2E
     20
120E 0D 0A              STRING_CRLF     DB      0Dh,0Ah


```

```
                                PAGE
                        STRING_1701     DB      "1701"
1210 31 37 30 31
                        STRING_A        DB      "-A"            ; Controller SRAM error
1214 2D 41
                        STRING_B        DB      "-B"            ; Controller diags error
1216 2D 42
                        STRING_C        DB      "-C"            ; Test ready timed out
1218 2D 43
                        STRING_D        DB      "-D"            ; Recalibrate failed
121A 2D 44
                        STRING_E        DB      "-E"            ; Reset failed
121C 2D 45

                        INT13H_TRIVIAL_OP_VECTOR:
121E
                                        DB      INT13H_OP_01_GET_DISK_STATUS            ; INT13H services that
121E 01
                                        DW      offset INT13H_01_GET_DISK_STATUS        ;  do not require
121F 0D70 R
                                        DB      INT13H_OP_08_GET_DRIVE_PARAMS           ;  talking to the disk
1221 08
                                        DW      offset INT13H_08_GET_DRIVE_PARAMS       ;  controller
1222 0D0C R

                        DMA_COMMAND_TABLE:
1224
                                        DB      OP_08_READ_SECTORS              ; These are commands that
1224 08
                                        DB      OP_0A_WRITE_SECTORS             ;  transmit data using DMA
1225 0A
                                        DB      OP_E5_READ_LONG
1226 E5
                                        DB      OP_E6_WRITE_LONG
1227 E6
                                        DB      OP_0E_READ_SECTOR_BUFFER
1228 0E
                                        DB      OP_0F_WRITE_SECTOR_BUFFER
1229 0F

                        INT13H_OP_TO_COMMAND_BYTE:
122A
                                        DB      OP_00_TEST_DRIVE_READY          ; 00h
122A 00
                                        DB      OP_00_TEST_DRIVE_READY          ; 01h
122B 00
                                        DB      OP_08_READ_SECTORS              ; 02h
122C 08
                                        DB      OP_0A_WRITE_SECTORS             ; 03h
122D 0A
                                        DB      OP_05_VERIFY_SECTORS            ; 04h
122E 05
                                        DB      OP_06_FORMAT_TRACK              ; 05h
122F 06
                                        DB      OP_07_FORMAT_BAD_TRACK          ; 06h
1230 07
                                        DB      OP_04_FORMAT_DRIVE              ; 07h
1231 04
                                        DB      OP_00_TEST_DRIVE_READY          ; 08h
1232 00
                                        DB      OP_0C_INIT_DRV_PARM             ; 09h
1233 0C
                                        DB      OP_E5_READ_LONG                 ; 0Ah
1234 E5
                                        DB      OP_E6_WRITE_LONG                ; 0Bh
1235 E6
                                        DB      OP_0B_SEEK                      ; 0Ch
1236 0B
                                        DB      OP_00_TEST_DRIVE_READY          ; 0Dh
1237 00
                                        DB      OP_0E_READ_SECTOR_BUFFER        ; 0Eh
1238 0E
                                        DB      OP_0F_WRITE_SECTOR_BUFFER       ; 0Fh
1239 0F
                                        DB      OP_00_TEST_DRIVE_READY          ; 10h
123A 00
                                        DB      OP_01_RECALIBRATE               ; 11h
123B 01
                                        DB      OP_E0_SECTOR_BUFFER_DIAG        ; 12h
123C E0
                                        DB      OP_E3_DRIVE_DIAG                ; 13h
123D E3
                                        DB      OP_E4_CTRL_DIAG                 ; 14h
123E E4

                        PADDING         DB      0DB1h dup(0FFh)
123F 0DB1 [
              FF
                  ]

                        VERSION_TAG     DB      "2793VG.10010688"
1FF0 32 37 39 33 56 47
     2E 31 30 30 31 30
     36 38 38

                        CHECKSUM        DB      94h
1FFF 94

                        ROM             ENDS

                                        END
```

```
;--------------------------------------------------------
;                                                       :
; This has been disassembled and commented manually on  :
; Sat Sep 11th 2021 in hope it will be useful to        :
; somebody. When assembled, it matches the              :
; Victor V86P Hard Drive Controller ROM versioned       :
; "2793VG.10010688" byte-for-byte.                      :
;                                                       :
; Please note that it is not free software and is       :
; subject to copyright.                                 :
;                                                       :
;------------------------------------------ SMOKE WEED --
```